RESEARCH REPORT 509
Sahibsingh A. Dudani, Anthony L. Luk, Jacqueline P. Stafsudd,
Carol S. Clark, and Bruce L. Bullock
JULY 1977

MODEL-BASED SCENE MATCHING

HUGHES
HUGHES AIRCRAFT COMPANY

HUGHES RESEARCH LABORATORIES • MALIBU

HUGHES
HUGHES AIRCRAFT COMPANY
RESEARCH LABORATORIES

MODEL-BASED SCENE MATCHING

RESEARCH REPORT NO. 509

Sahibsingh A. Dudani
Anthony L. Luk, Jacqueline P. Stafsudd, Carol S. Clark
Bruce L. Bullock

Hughes Research Laboratories
3011 Malibu Canyon Road
Malibu, California 90265

ACKNOWLEDGMENTS

The authors wish to thank Professor Ramakant Nevatia of the University of Southern California for his valuable contributions towards the development of concepts relating to the region models. The authors gratefully acknowledge Dr. Joe Jenney (now with the Defense Advanced Research Project Agency, Washington, D.C.) for his many helpful suggestions and contributions made in the initial phase of this study. The direction and encouragement provided by Dr. Don Close and Dr. Charles McNary have been largely responsible for the progress of this project. The authors also wish to thank Mr. Gary Klein, who derived the relationships for the least-squares straight-line fit. This work was supported by DARPA Contract F30602-77-C-0049, and monitored by the Rome Air Development Center, New York.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1

INTRODUCTION

The problem of matching two scenes is very important in many practical applications. This report deals with the development of matching techniques useful for image-based guidance systems in remotely piloted vehicles and missiles. This is a particularly difficult matching problem because the outdoor scenes encountered have an almost unlimited variety of scene content in an uncontrolled and varying environment. The present state of image analysis technology does not permit a totally general solution to the scene-matching problem. The philosophy of our approach to scene matching has been to narrow the problem somewhat by using only a few types of key scene features for the matching process. However, we do consider the problem from a fairly general standpoint in the sense of not limiting the contents and other variables associated with the sensed scene to be matched. Our efforts in this area are directed towards developing an automatic, image-based guidance system within the near-term.

The general problem of scene matching may be defined as follows. Let there be two given scenes, one called a reference (template) scene, f, and the other called a sensed (test) scene, g. The reference scene is generally acquired ahead of time and then used as a template or a mask for the matching process. On the other hand, the sensed scene is acquired live and then matched against the reference. In most applications, the reference scene is much smaller than the sensed scene. The matching process locates the portion of the sensed scene g that best matches the template scene f. Since an exact geometrical match does not always exist between the two scenes observed from different ranges and perspectives, results are provided in terms of pairs of matching reference and sensed points (often referred to as control points). The necessary navigational information for guidance can be derived from these control points.

Our approach to scene matching is based on the use of scene models. A scene model is a data structure composed of feature

descriptors, their placement and spatial relationships in the scene. A model is constructed ahead of time from the reference scene and stored for the matching process. Since this model includes only the most prominent features, the likelihood of finding the same features in the scene is greater. For the sensed scene, a model is constructed on-board in a completely automatic, non-interactive mode. This model is more complex than the reference model because there is less freedom in selecting features. The matching process compares the two models and arrives at the control points between the two scenes, which are then used to derive the necessary navigational information. Figure 1 shows the different steps for a model-matching scheme used in an image-based guidance system.

This report presents the current status at the Hughes Research Laboratories (HRL) in the area of model-based scene matching. Details of many different techniques used in building and matching scene models are described in this report. Areas of the complete matching system which need improvement are identified. An important aspect of assessing image-analysis techniques is their sensitivity to the parameters used; this will be addressed in this report. Some new techniques are also recommended for future work to improve model-matching performance.

Section 2 briefly compares the model-based schemes to the correlation techniques traditionally used for scene matching. Section 3 deals with the problem of feature selection in building the scene models. We are currently using two basic scene models – vertex and region. The vertex model is a simplified representation of a scene based on the presence of straight-line edge segments. It is most useful where image resolution is high enough that fine structural details (particularly those of man-made structures) are visible. For the past two years we have concentrated mainly on developing and implementing such a scene model. Sections 4 and 5 describe, respectively, the techniques being used for vertex model construction and matching. We recently began investigating region models, which represent a scene by using large, key areas which have uniform properties (such as intensity or

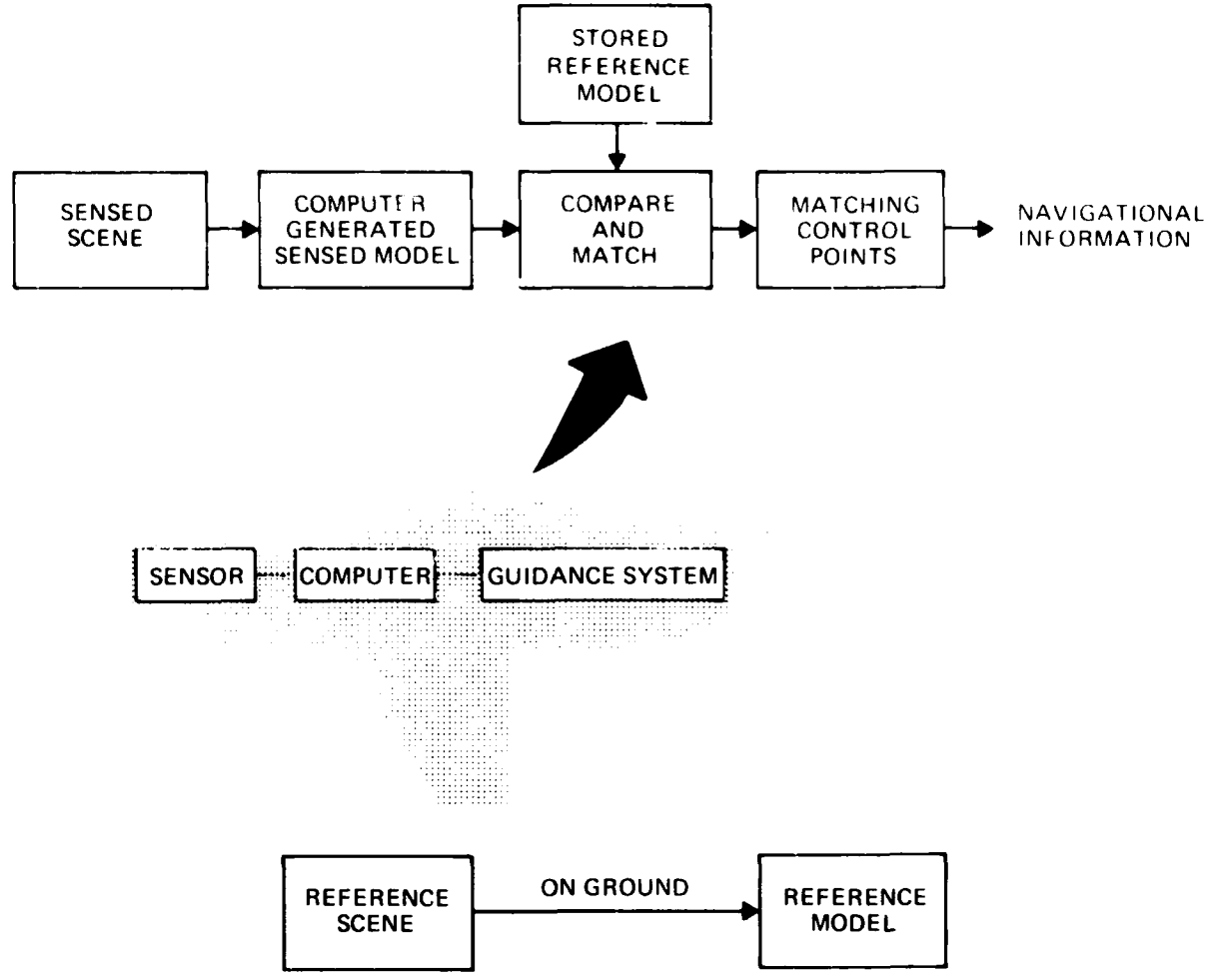


Figure 1. Model-based scene matching in a guidance system.

texture). This type of scene model is most useful in situations where only the gross geometric features are resolvable. Section 6 discusses our preliminary results on the use of such scene models. Some conclusions and plans for future work are discussed in the last section of this report.

SECTION 2

CORRELATION AND MODEL-BASED SCHEMES

There are two basic approaches to scene matching: cross-correlation and model-based schemes. Cross-correlation techniques have received the most attention because of their conceptual simplicity. However, they do have some inherent difficulties and limitations. Model-based schemes, on the other hand, are complex but lack inherent limitations. Both approaches are briefly described below.

A. AREA CROSS-CORRELATION

A common approach to matching two scenes is to use the cross-correlation coefficient.[1] This measure is proportional to the root-mean-square difference between the corresponding pixel intensities for the two scenes. The cross-correlation coefficient computes the common energy between the two scenes as a function of the position of the reference scene within the bounds of the sensed scene. In this matching process, the maximum value of the correlation coefficient indicates the best match position between the reference and the sensed scene. As a modification to the basic technique, the correlation may also be performed on the intensity gradients.

In practice, the position indicated by the maximum correlation coefficient is often found to be an incorrect match between the reference and the sensed scene. This effect, commonly known as false-fixing, may occur even though the two scenes contain an amount of unique geometrical signature normally considered adequate for reliable matching. This problem of improper scene registration may be due to differences between the reference and the sensed scenes in contrast level, range (scale), viewing angle, context (missing or additional parts), imaging sensor type, and seasonal factors. To use correlation successfully in such situations often requires transforming the reference scene to compensate for some of the differences between the two scenes being matched. However, it is often difficult in practice to

represent and implement the necessary transformations. In addition, to perform these transformations requires enormous computational power. For some of these situations, multiple references may be stored to compensate for differences such as viewing angle and context. In those cases, the memory requirement for storing the necessary number of references may also be enormous. However, the correlation approach works very well in situations where the reference and sensed scenes have been obtained only a few time frames apart. In such cases, the scene differences discussed above are minimal; a classic example of this is the use of correlation in image-based trackers.

B. MODEL-BASED SCHEMES

The problem of cross-correlation failure, which is discussed above, results from differences between the reference and the sensed images. For the matching process to be less sensitive to these differences, the reference and sensed models must use invariant features based on some geometric data or description. The changes in the geometrical features caused by differences in range, viewing angle, illumination, sensor type, etc. are not as dramatic as on the pixel intensities. Thus, using such features in matching schemes avoids many of the difficulties present in correlation techniques.

A basic scene model may be defined as a collection of geometric features and their relationships in the scene. The general form of the model is a compact data structure or graph[2] which consists of nodes and links between the nodes. The feature names become labels on the nodes, and their relationships become labels on the links in the graph. The matching techniques use the graph structures from the reference and the sensed scene models. The reference model is usually constructed ahead of time and stored; in contrast, the sensed model is constructed on-line from the sensed scene.

Model-based scene matching, by using geometric features, provides a flexible way to deal with a wide range of image differences. Where there are large differences in viewing angles, it may still be necessary

for this matching technique to store multiple reference models. However, the reference generation process would still be much simpler and have greatly reduced memory requirements. This process is also a convenient way of incorporating information about contextual and other image variations. One useful charactersitic of model-based matching is that it allows finding a partial match between the reference and the sensed scenes in situations where only a small part of the reference image has a match with a certain part of the sensed image. Finding such a partial match can be very difficult with a correlation-matching approach.

SECTION 3

FEATURE TYPES FOR SCENE MODELS

Image features fall into three general categories: point, local, and global. Figure 2 shows, for each category, an example of a simple scene and its representation. The trade-offs between these features are discussed below.

A. POINT FEATURES

Point features are used to represent a scene as a matrix of values for every resolution element or pixel in the image. The point value for each pixel represents either the intensity magnitude (a function of the reflectivity or emissivity) or the range from the sensor to the point. Figure 2 shows a matrix of values which represent the pixel intensities. Point measures have the advantage that

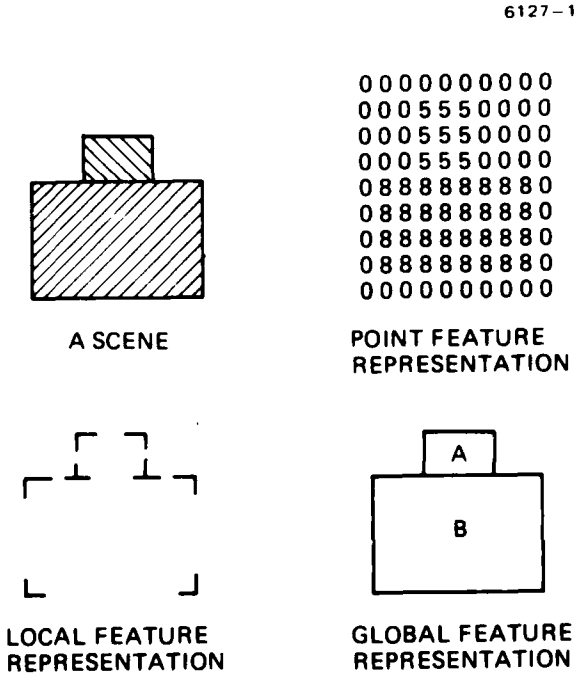


Figure 2. Scene representation through the point, local, and global types of features.

they are usually available directly from the image sensors with little additional processing required for their extraction. Their major disadvantage is that they have poor invariance characteristics. For example, they can vary widely with small changes in illumination and contrast levels. It is sometimes possible to perform transformations on the point feature data to overcome the lack of invariance but, as a rule, these transformations are computationally very complex. The traditional correlation-matching techniques generally use point feature image representations.

B. LOCAL FEATURES

Local-feature measures include average intensity over an area, texture properties over an area, locally connected line segments and curves, and line and curve intersections. Features based on local measures have greatly improved invariance characteristics in comparison with point measures. These invariance characteristics arise from local averaging and the use of relative measures, as in the detection of edges. The presence of a line segment, for example, will not change for a wide range of illumination and viewing angle changes, even though the absolute values of the point features producing the gradient may shift dramatically.

The point features represent an image exactly, although with little invariance or data compression efficiency. Local features, on the other hand, represent an image in an abbreviated or abstract manner. The relative positions and orientations of line segments and line intersections, for example, may be sufficient to specify an object's shape. A model using local features for image matching has the advantage of greater invariance to image differences and a smaller memory requirement compared to that for point feature matching. An example of a local feature model is also shown in Figure 2. In this example, the local features are corners.

C. GLOBAL FEATURES

Global features include regions, entire surfaces, shapes, and objects that have been segmented or extracted from an image. A global representation or model for a building might consist of several rectangles connected in a specific way. A trivial global representation of a block structure with two separate regions A and B is shown in Figure 2.

Global features have a high degree of invariance to image differences. Unfortunately, global features are the most difficult to extract successfully. This is because they depend on the segmentation of complete regions or surfaces from the scene.

Table 1 summarizes the above discussion on image features. This table shows that the point features suffer from poor invariance characteristics and, therefore, are inappropriate for use in scene models. We have considered two different types of scene models — those based on local features and those based on global features. The vertex model uses local features such as edge segments and their end points and intersections. This model is most useful when fine structural details are present in the scene. The vertex model can also be used in some long-range scenes which contain roads or large structures with straight-line geometrical characteristics. A detailed description of vertex models is given in Section 4. The region model is based on global features such as large key areas with uniform intensity or texture. This model is most useful in long-range scenes with regions such as rivers, lakes, roads, and mountain areas. Building a region model is relatively more difficult, since it is necessary to use segmentation techniques to extract uniform areas. Most segmentation techniques reported in the literature are complex and have not yet been perfected. Section 6 describes our initial work on construction and matching of these models.

Table 1. Comparison of Feature Types for Scene Models

6127-3

CATEGORY	INVARIANCE	EXTRACTION DIFFICULTY	TRANSFORMS TO CORRECT FOR INVARIANCE ERRORS
POINT FEATURES	POOR	TRIVIAL	DIFFICULT
LOCAL FEATURES	GOOD	MODERATE	NOT ALWAYS NECESSARY
GLOBAL FEATURES	EXCELLENT	DIFFICULT	NOT NECESSARY

SECTION 4

SCENE REPRESENTATION THROUGH VERTEX MODELS

The vertex model is a scene representation based on geometrical and structural features with straight-line characteristics. This model is appropriate for the class of scenes containing some prominent straight-line edge segments. It is useful in many applications, since man-made structures often contain such straight-line geometrical features. The straight-line features in a scene are quite robust since they are largely insensitive to several image variations (such as illumination, viewing angle, and sensor type).

A vertex model consists of data structures which represent the position of vertices and their interconnections as observed in the scene. A vertex as defined here may be either (1) a common point for two or more straight lines or (2) an end-point of a straight line. There are two main steps in constructing vertex models: (1) locating straight-line edge segments and (2) computing vertex positions and their interconnections from the straight-line segments found. The block diagram in Figure 3 illustrates the sequential steps for constructing the vertex model. Each step is described in this section. A simple scene containing some blocks will be used to illustrate the techniques for the different steps in this process. Experimental results for vertex model representations of several outdoor scenes will also be considered.

A. EDGE DETECTION

Many edge-detection methods described in the literature apply a local operator to points in a picture to obtain short edge elements. Some of these operators are relatively simple, such as Roberts and Sobel,[3] and some are more complex, such as Hueckel.[4] The Roberts and Sobel operators first compute gradients in x and y directions, and

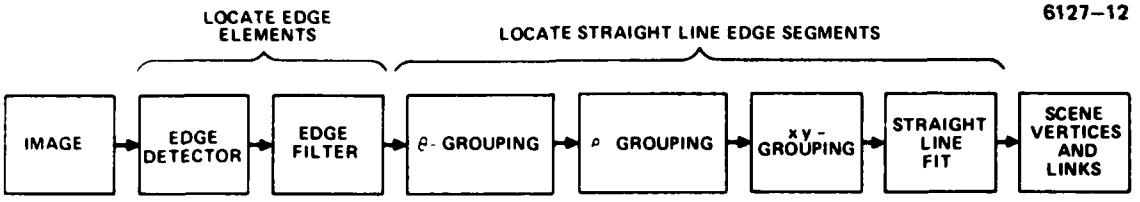


Figure 3. Block diagram for vertex model construction from an image.

then from these the edge magnitude and direction are computed. The computations involved in these operators are shown below:

- Roberts edge detection*:

A	B
C	D

$\Delta x = (B+D) - (A+C)$

$\Delta y = (C+D) - (A+B)$

Edge magnitude at point A
$= |\Delta x| + |\Delta y|$

Edge direction at point A
$= \tan^{-1} \frac{\Delta y}{\Delta x}$

- Sobel edge detection:

A	B	C
D	X	E
F	G	H

$\Delta x = (C+2E+H) - (A+2D+F)$

$\Delta y = (F+2G+H) - (A+2B+C)$

Edge magnitude at point X
$= |\Delta x| + |\Delta y|$

Edge direction at point X
$= \tan^{-1} \frac{\Delta y}{\Delta x}$

If the magnitude of the computed edge is above the chosen threshold, an edge element is declared at the pixel. There are two main problems in using these simple edge-detection schemes: (1) selection of the threshold and its sensitivity to the total number of edges and (2) multiple edge detection (i.e., several edge elements are detected around the point where an edge may ideally be expected). If the threshold value is not selected properly, there may be either too many or too few edge elements. To use these edge detectors successfully requires using some sort of adaptive method of arriving at the appropriate threshold. The method perhaps could be based on the underlying goal or application. These edge detectors often give "noisy" edges because

*This is a modification of the original Roberts edge operator.

of the way in which they operate locally on pixel values. Also, the degree of accuracy in estimating edge element orientation is not very high in these operators.

An example of an edge detector which is more complex, but less sensitive to local noise is the Hueckel operator. The edge model used by this operator is a step function F in a circular disk. Let

$$F(x,y,c,s,\rho,b,d) = \begin{cases} b & cx + sy \leq \rho \\ b + d & cx + sy > \rho, \end{cases}$$

where the origin of the x-y coordinate system is at the center of the circular region. The intensity levels on the two sides of the step edge are b and b + d. The equation of the straight-line edge is given by $cx + sy = \rho$. Figure 4 shows a circular Hueckel disc with a step edge. Let E(x,y) be the given intensity function in the circular disc. The intensity function E is approximated with an ideal step edge F so that the distance between E and F is minimum i.e., we minimize

$$\iint [E(x,y) - F(x,y,c,s,\rho,b,d)]^2 \, dx \, dy.$$

The minimization procedure returns both the best edge position (x_e,y_e) and its orientation. There are two parameters associated with this operator: (1) DIFF, which is related to the step edge size, and (2) CONF, which is a threshold value for the edge confidence measure. Our experience using this operator has shown that values of 100 and 0.85 were found to be suitable for DIFF and CONF, respectively, for 8 bit grey-level pictures. The edges found are relatively less sensitive to the exact values of these parameters. This is a major advantage of using the Hueckel operator. Also, this operator is to a high degree insensitive to local noise. The disadvantages of this operator are that it is complex and that it uses an ideal step edge model. In an outdoor scene, an ideal step edge model often misses some important edge elements.

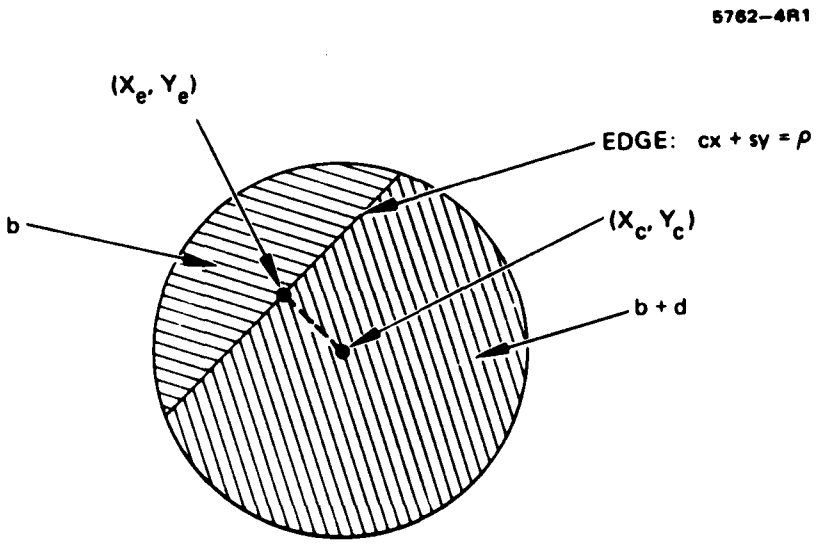


Figure 4. Hueckel edge model consisting of a step edge in a circular disc.

To avoid the problem of obtaining multiple edge elements with the Hueckel operator, an additional requirement is imposed on edge element acceptance. An edge element (x_e,y_e) is accepted only if

$$|x_e - x_c| \leq 1 ,$$

and

$$|y_e - y_c| \leq 1 ,$$

where (x_c,y_c) is the center of the Hueckel disk. An edge element satisfying this condition is referred to as a central edge. An example of imposing this additional requirement on edge detection is shown in Figure 5. In this figure, note that multiple edge points were present before the additional condition was imposed. The processing for extracting edge segments becomes simpler if only the central edges are retained. The result of edge detection on a given scene is a list of edge elements which are defined by their x and y coordinates and their orientation.

In most of our work, we have used the Hueckel operator for the edge-detection step. However, because of its complexity and tendency to miss edges in outdoor scenes, we are considering alternative edge-detection operators. To use the Roberts or Sobel operator effectively, a technique for adaptive threshold computation must first be developed. In addition to these operators, a few other heuristic edge-detection techniques will be evaluated on outdoor scenes appropriate to our application.

B. EDGE FILTERING

The edge-detection process on a given scene produces a list of edge elements specified by their position and orientation. The edge element is a basic unit for the remaining processing necessary to construct the vertex model. The success or failure of the complete process depends on the information derived from these edge elements.

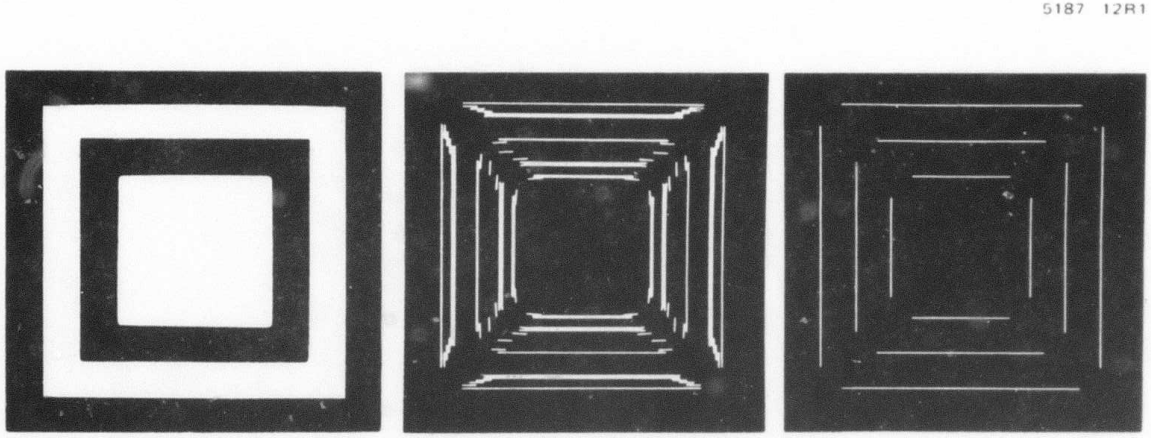


Figure 5. Multiple and central Hueckel edge elements for a simple picture.

The edge elements obtained may be grouped into two separate classes: undesired edges and desired edges. The undesired edges may be produced by sensor noise, use of an improper edge model, and the curved structures in the scene. Edge elements resulting from sensor noise are distributed randomly throughout the scene. As an example of undesired edges resulting from an improper edge model, the Hueckel operator gives false edges at corners or intersections of straight-line segments. The bulk of the undesired and the desired class of edge elements come from the curved and the straight-line scene structures.

The major step (after edge-element detection) in the vertex model construction is the extraction of straight-line edge segments. An edge segment is formed by grouping edge elements which form a contiguous part of a straight line. Because we are only interested in edge elements which form straight-line segments, a filtering technique is used to eliminate edge elements which do not contribute towards formation of straight-line edge segments. There are two main payoffs from edge filtering of this type: (1) reduced data for further processing and (2) increased chances of finding short line segments by working with cleaner data. It is important that the filtering process retains as many edge elements belonging to short edge segments as possible.

A straight-line edge filter, as we define it, retains an edge element (x, y, θ) if and only if a minimum total of n edge elements with angles in the range $(\theta \pm 15^{o})$ are found within a rectangular window of width 1 and length ℓ pixels centered around (x,y) and in the direction θ as shown in Figure 6. Each element found in the edge detection process is checked to see if it satisfies this edge filter requirement; if it does not, the edge is rejected.

The values of the two parameters n and ℓ for the straight-line filter must be preselected. From the definition, we know that the number of edge elements retained after filtering is directly proportional to ℓ but inversely proportional to n. An experiment was performed on several outdoor scenes to examine the relationships between number of edges retained and n and ℓ. An example of the experimental results obtained is shown in Figure 7. Note that the edge elements

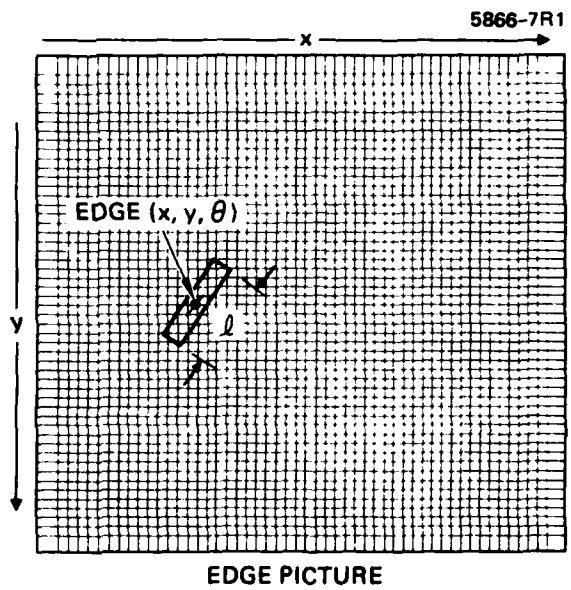
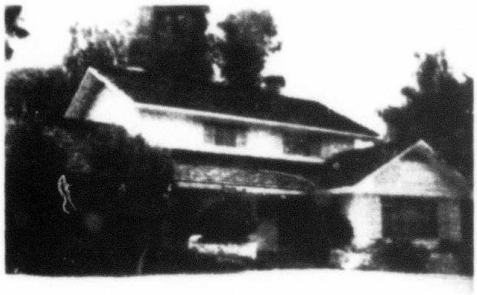
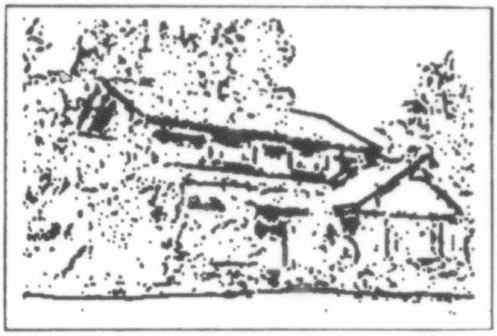


Figure 6. Filter window centered around an edge element.

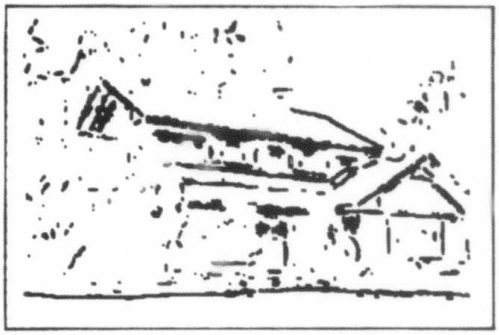
5866-9R1



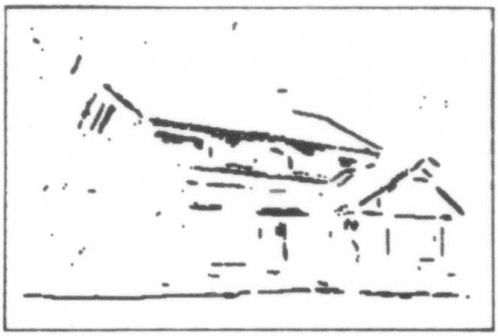
(a) SCENE (360 x 240)



(b) EDGE PICTURE:
6317 EDGE ELEMENTS



(c) FILTERED PICTURE:
3972 EDGE ELEMENTS (63%)
n = 3 and ℓ = 11



(d) FILTERED PICTURE:
1657 EDGE ELEMENTS (26%)
n = 7 and ℓ = 11

Figure 7. Straight-line edge filtering on an outdoor scene containing structures with straight-line characteristics and natural background (such as bushes and trees).

found are indicated by point intensities. Two cases of edge filtering with different parameters are also shown in the figure. In the first case, with parameters n = 3 and ℓ = 11, the majority of edges belonging to bushes and trees are filtered out and most of the edges belonging to straight-line segments (including short ones) are retained. In the second case, the value for n is increased from 3 to 7 with ℓ remaining at 11. In this case, almost all edges belonging to natural background are lost, but a significant portion of the edge elements belonging to straight-line segments are also lost. Only 26% of the total edge elements are retained in this case.

For the same example, relationships for edges retained with n for three different values of ℓ are shown in Figure 8. Note that the results of the filtering process are similar for the window lengths of 7, 9, and 11 when n = 2. As n increases, the results obtained for the three window lengths differ significantly.

To extract all short and long straight-line edge segments from a scene, the parameters for the filtering process should be chosen such that edge elements are rejected only if they have almost no correlation with its neighboring edge elements. This is achieved by setting the value of n relatively low. From our experience in processing outdoor scenes, we have found values of 2 and 7 for n and ℓ, respectively, to

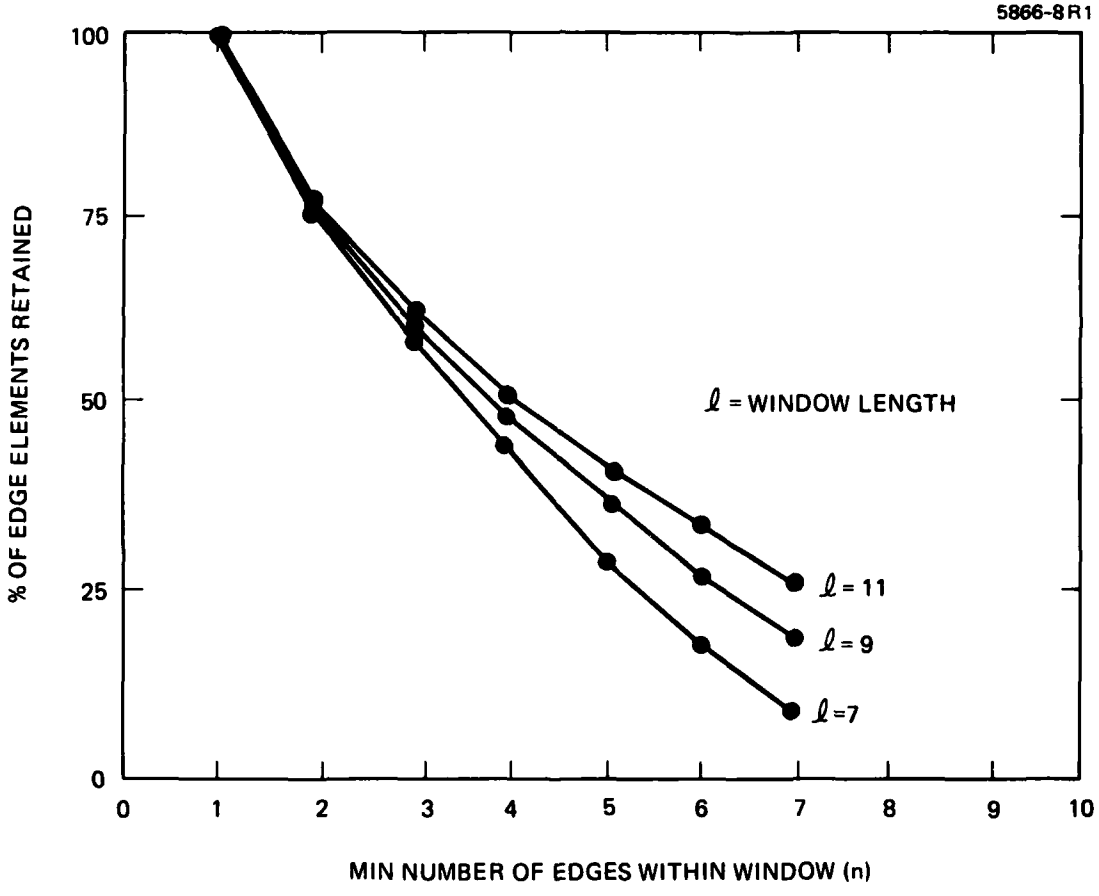


Figure 8. Relationships of number of edge elements retained with minimum number of edges within the filter window (n) and the window length (ℓ).

produce good results in terms of retaining edge elements belonging to short straight-line edge segments. Also the computational requirements with these values of n and ℓ are reasonably low. As ℓ is increased, the computation time used by the filtering process increases more than linearly.

When there is a hierarchical program controlling the complete edge-segment extraction, the values of n and ℓ can be chosen on the basis of the particular goals. For example, if, as a first step, only the long and prominent edge segments were to be extracted, the values of n should be chosen relatively high to remove the edge elements belonging to short segments. At present, the vertex model construction process is not driven through a hierarchical program, and therefore the values for n and ℓ are preselected.

C. STRAIGHT-LINE EDGE-SEGMENT EXTRACTION

The list of edge elements $\{(x_i, y_i, \theta_i), i = 1, N\}$ obtained by edge detection and filtering is used in extracting straight-line edge segments. The main problem is how to fit straight-lines to the edge elements located. To do this requires dividing the entire edge element list into several groups, each representing a single contiguous line segment. Once this is done, the next step, which is to fit a single straight line to each group and to determine the two end points of the line segment, is quite simple. The final result is a list of line segments represented by the x and y coordinates of the two end points of each segment. Figure 9 illustrates the steps involved in dividing the entire edge element list into several groups and the subsequent extraction of line segments. These steps are described below.

1. A Parametric Representation of Straight Lines

A straight line may be represented parametrically by the following relationship[5]:

$$\rho = x \cos \theta + y \sin \theta ,$$

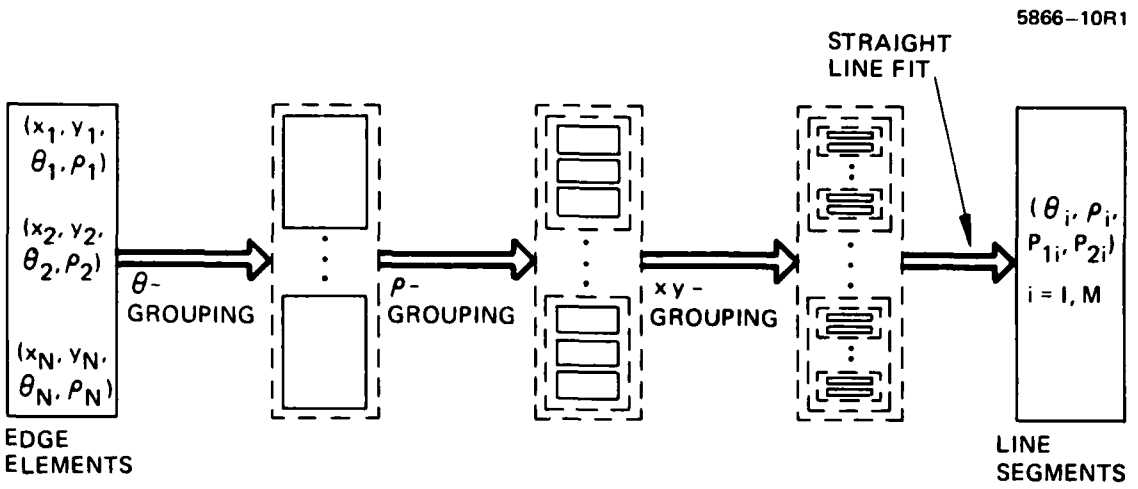


Figure 9. Extracting straight-line segments from a list of edge elements.

where θ is the angle of the intensity vector to the given line and ρ is its displacement from the origin (as shown in Figure 10). The intensity vector is defined as the normal to the given line directed from the low grey-level to the high grey-level side. The value of ρ is considered positive when the intensity vector points away from the origin and negative when it points toward the origin. This θ-ρ representation has the advantage over the conventional slope-intercept ($y = mx + c$) line representation that the values of θ and ρ lie within closed intervals:

$$0^0 \leq \theta \leq 360^0$$

$$|\rho| \leq (X_{max}^2 + Y_{max}^2)^{1/2}$$

where X_{max} and Y_{max} are picture width and height, respectively. The techniques described here for extracting line segments use this θ-ρ representation for straight lines.

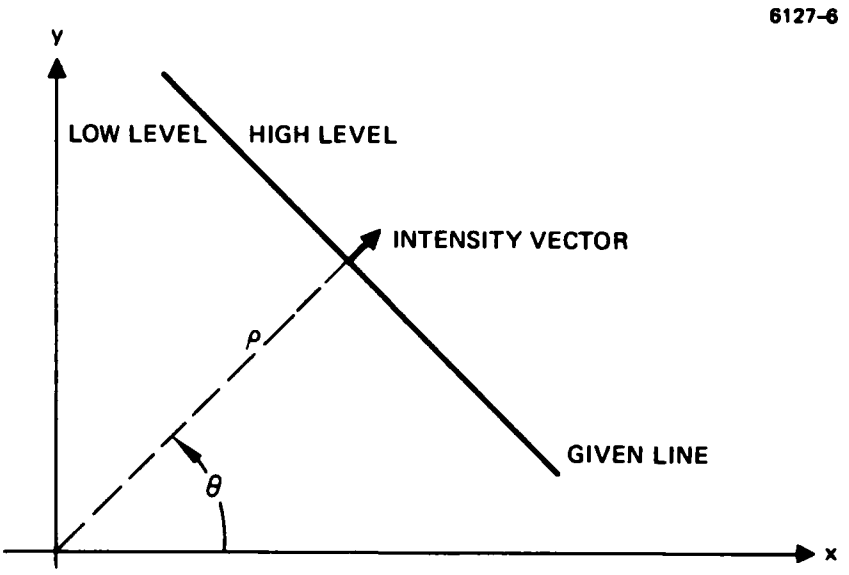


Figure 10. A parametric representation of straight lines.

2. θ-Grouping

Let $\{(x_i, y_i, \theta_i), i = 1, N\}$ be the position and direction (intensity vector angle as defined in Section 4.C.1) of the edge elements found. The θ-distribution of the edge elements, which gives the number of edge elements present having a given angle θ, is computed from the edge element list. A scene containing some simple block structures is used here to illustrate the method of dividing edge elements into line segment groups. Figure 11 shows the original scene and the edge elements found. Because of the simplicity of the scene being considered, no edge filtering is necessary. The θ-distribution for this example is shown in Figure 12(a). The θ-distribution is a closed graph in the sense that 0^o and 360^o represent the same edge element direction.

As described earlier, the main problem in finding line segments is the division of the edge element list into several groups, each representing a single line segment. As a first step, the edge elements

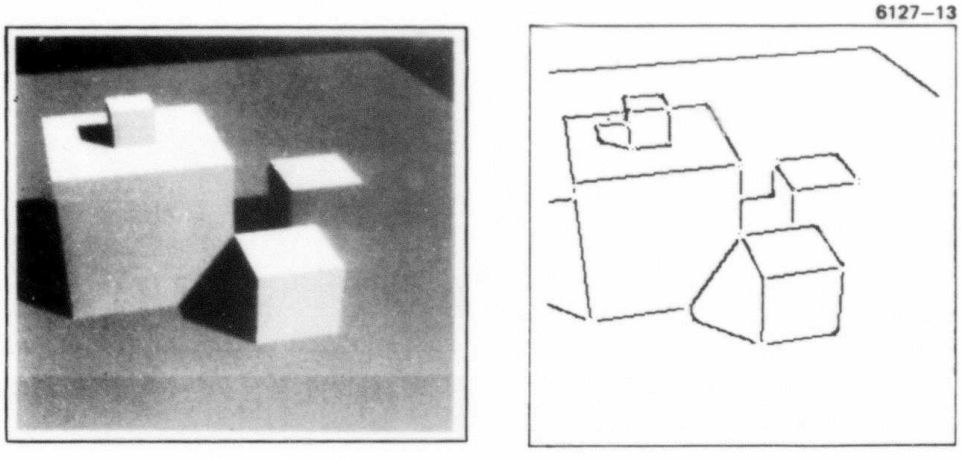
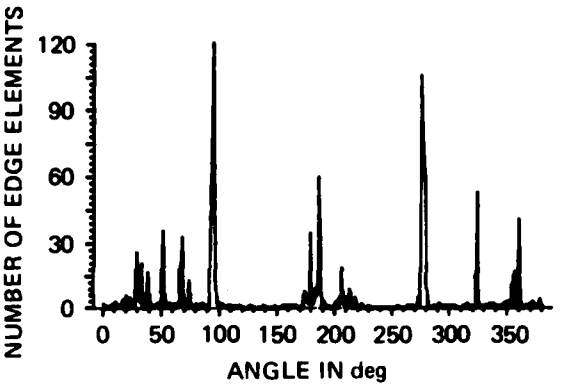
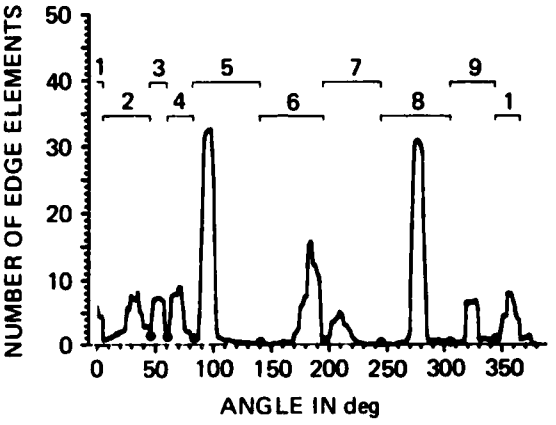


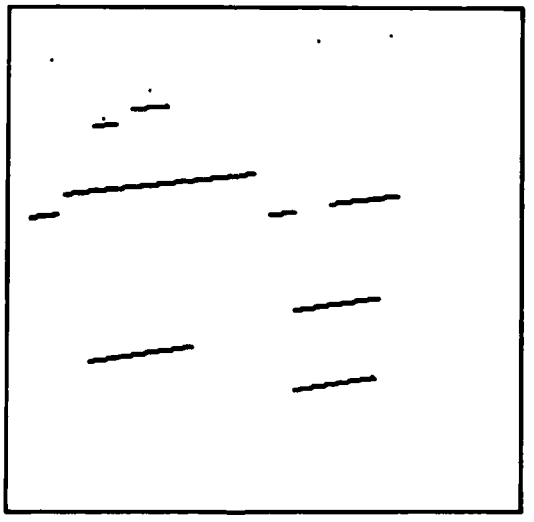
Figure 11. Hueckel edge detection on a scene containing simple block structures.



(a) θ-DISTRIBUTION



(b) GROUPING ON AVERAGED θ-DISTRIBUTION
(●) LOCAL MINIMUM POINT



(c) EDGE ELEMENTS FOR θ-GROUP NO. 8 (249^{o} - 309^{o})

Figure 12. θ-grouping on scene of Figure 11.

are broken by dividing the complete θ-range (0° to 360°) into several divisions. This is done in two steps.

First, a smooth θ-distribution is obtained by local averaging the original distribution through the use of a moving window. Let the width of the moving window be w. To have a symmetrically centered window around a point, the value selected for w must be odd. Note that a value of 1 for w has no smoothing effect, since the original distribution is unaltered. The value of w is chosen such that a reasonably smooth distribution is obtained. For most outdoor scenes, a value of 11° for w appears to give acceptable results. Figure 12(b) shows the averaged θ-distribution for w=11°.

Second, the θ-range is partitioned where there are local minima in the averaged θ-distribution. A point in the distribution is considered to be a local minimum only if it is less than all other points in a local neighborhood. If there is a local plateau present in the distribution, the local minimum point is considered to be at the middle point of the plateau. A moving window is used to define the local neighborhood for locating minima. As a rule of thumb, the size of the local window is chosen to be of the same size as the averaging window used for obtaining a smooth θ-distribution. The partitioning of the θ-range is done at the local minimum points so that the likelihood of assigning edge elements belonging to a single line segment into two different groups is small. Each θ-group obtained in this manner includes edge elements belonging to line segments having approximately the same direction. For the example under consideration, a total of nine θ-groups were obtained through this procedure. The θ-range and the number of edge elements for the groups found are given in Table 2. Figure 12(c) shows edge elements belonging to the θ group No. 8, which has a range of 249° to 309°.

3. ρ-Grouping

This step considers each θ-group separately and attempts to divide it further into smaller groups based on the ρ parameters of the

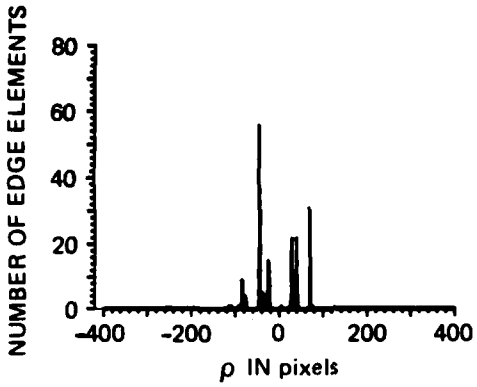
Table 2. θ groups for the Block Scene of Figure 11.

θ Group No.	θ range, deg	Number of Edge Elements
1	1 to 6, 341 to 360	99
2	7 to 46	139
3	47 to 60	84
4	61 to 87	115
5	88 to 140	377
6	141 to 195	206
7	196 to 248	76
8	249 to 309	358
9	310 to 340	85

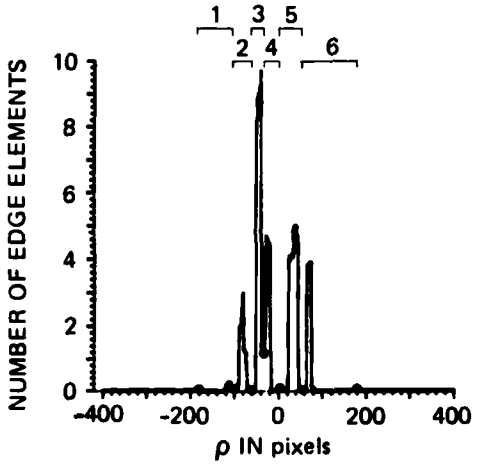
edge elements in the θ group. For each edge element in the group, ρ is computed through the relationship

$$\rho = x \cos \theta + y \sin \theta \ .$$

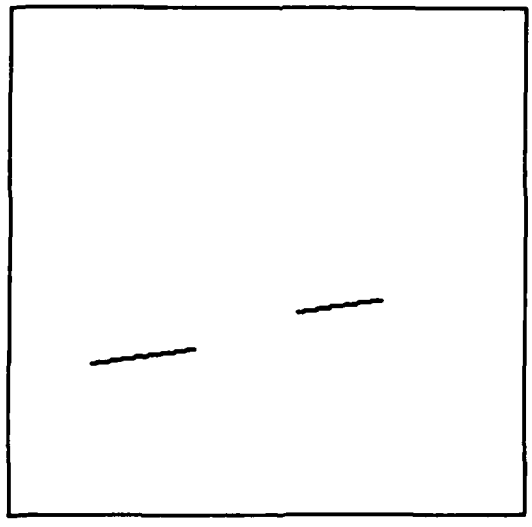
For each θ-group, a ρ-distribution is first obtained. This distribution gives the number of edge elements having a given value of parameter ρ. The partitioning of the ρ-range is also performed in a manner similar to that for the θ-range. Figure 13 shows the original and averaged ρ-distribution for the θ-group No. 8 shown in Figure 12(c). In this case the averaging on the ρ-distribution was performed using a window of 11 pixels wide. This value has provided good results on a variety of outdoor scenes. This procedure is not sensitive to the exact width of the averaging window. All that is needed is a reasonably smooth distribution adequate for locating local minima. Next, local minima are detected on the averaged ρ-distribution with a local window width of 11 pixels. In this example, seven local minima points are obtained, which gives rise to six different ρ-groups. The ρ-range and the number of edge elements for the groups obtained are given in



(a) ρ -DISTRIBUTION



(b) GROUPING ON AVERAGED ρ-DISTRIBUTION
(●) LOCAL MINIMUM POINT



(c) EDGE ELEMENTS FOR ρ -GROUP NO. 5 (3 TO 56)

Figure 13. ρ-grouping on θ-group No. 8 corresponding to Figure 11.

Table 3. Figure 13(c) shows the edge elements belonging to ρ group No. 5, which has a range of 3 to 56 pixels.

4. xy-Grouping

At the end of ρ-grouping, the original list of edge elements for the entire scene has been divided into several subgroups, the edge elements of which have approximately the same θ and ρ values. An example of such a subgroup was shown in Figure 13(c). This step considers edge elements in each θ-and ρ-group spatially (x-y space) and examines whether further grouping is possible. The idea is that groups of edge elements are to be obtained to which a single contiguous line may be fit. This procedure, which is similar to what is commonly referred to as point clustering in x-y space, involves the following steps:

- The first edge element in the group is considered to form subgroup (cluster) No. 1.
- The rest of the edge elements in the group are considered one by one. Each is checked to see if it lies within a preselected threshold distance (usually about 5 pixels) from the elements already considered. If the element under consideration does lie within the threshold distance from another neighboring element, it is included in the subgroup to which its neighbor belongs. Note that group merging is performed when an edge element is within the threshold from two separate groups.
- If, however, the edge element under consideration does not lie within the threshold distance from any other edge element, then it is considered to form a new subgroup or cluster.

Any subgroups containing fewer than a preselected threshold number of edge elements are eliminated at this step. This may be considered to be a filtering operation to disregard short edge segments. A simple example of xy-grouping is shown in Figure 14(a), where θ-ρ group is further divided into two subgroups, each representing a contiguous line segment.

Table 3. ρ Groups for the θ-group No. 8 Belonging to the Block Scene of Figure 11.

ρ Group No.	ρ Range, Pixels	Number of Edge Elements
1	-181 to -101	3
2	-100 to -61	36
3	-60 to -32	116
4	-31 to +2	54
5	3 to 56	105
6	57 to 181	44

5. Straight-Line Fitting

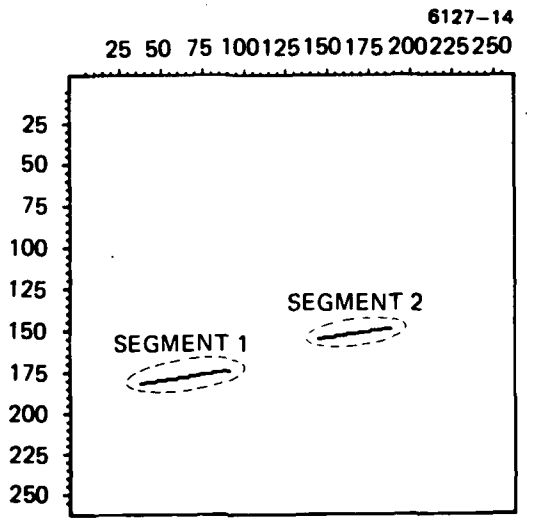
The three grouping steps — θ, ρ, and xy — produce several subgroups, each of which contains edge elements representing a single line segment. Next, a straight line is fit to the edge elements in each of the subgroups, and the two end points of the line segment are determined. In determining the best-fit straight line, only the x-y position information of the edge elements is used. This is advantageous in a way as it is not necessary to estimate edge element directions very accurately, and therefore simple and less accurate edge detection and orientation techniques may be used.

The criteria used for determining the best-fit straight line is to minimize the total normal distance from edge elements to the straight line. The normal distance from a point (x_i,y_i) to the straight line $(x \cos \theta + y \sin \theta = \rho)$ is given by:

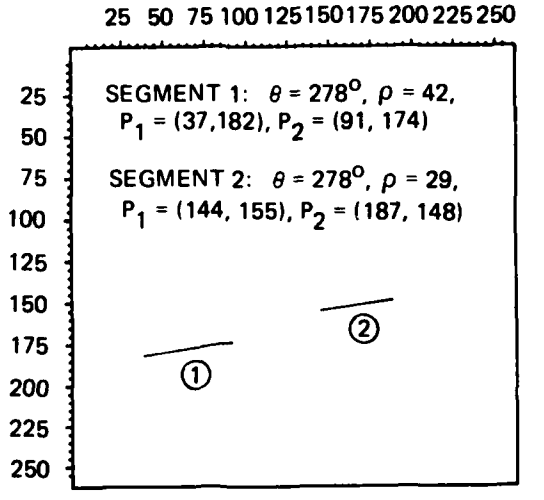
$$d_i = \rho - x_i \cos \theta - y_i \sin \theta \ .$$

The total normal distance D for the N points is given by:

$$D^2 = \sum_{i=1}^{N} (\rho - x_i \cos \theta - y_i \sin \theta)^2 \ .$$



(a) x y - GROUPING
(θ-ρ-GROUP: 249° - 309°, 3 - 56 pixels)



(b) FITTING STRAIGHT LINES

Figure 14. xy-grouping on edge elements of Figure 13.

The best-fit straight line (θ,ρ) is determined by solving the following simulaneous equations which minimize D^2.

$$\frac{\partial D^2}{\partial \theta} = 0$$

$$\frac{\partial D^2}{\partial \rho} = 0 \ .$$

A solution to these equations is presented in Appendix A. Appendix A also includes the relationships used for determining the end points of the line segment.

There are two measures of goodness of fit that may be used for selecting or filtering line segments. The first is the root-mean-square (rms) distance from a point to the line and is given by:

$$\bar{d} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\rho - x_i \cos \theta - y_i \sin \theta)^2} \ .$$

A value of zero indicates an ideal fit. The acceptance value of this measure depends on the resolution of the scene being considered. In our work on outdoor scenes, we have chosen 2 pixels as the acceptance value. The second is the number of points constituting the edge segment divided by the length of the line:

$$\eta = \frac{N}{\ell c} \ ,$$

where ℓ is the length of the line segment and c is a compensating factor based on the slope of the line. The compensating factor c is given by:

$$c = \cos \theta \qquad \text{if} \qquad |\tan \theta| \leq 1$$

$$= \sin \theta \qquad \text{if} \qquad |\tan \theta| > 1 \ .$$

Figure 14(b) shows the straight-line fit to the line segment subgroups of Figure 14(a). The results of line-segment extraction on the complete block scene are illustrated in the next section.

D. VERTEX MODEL CONSTRUCTION

The last section described the procedure for extracting straight-line edge segments from a scene. The line segments found are represented by their θ-ρ parameters and the two end points. The purpose of this step is to locate vertices based on the line segments found. A vertex as defined here may be either (1) a common point for two or more straight lines or (2) an end point of a straight line. The positions of vertices and the interconnection between them are determined to form the complete scene vertex model.

For each line end point, a search is made to check if any other end points or line segments are present within its circular neighborhood. Our experience indicates that a neighborhood of about 5 pixels in radius is generally adequate. This is because the edge detection process often fails to find proper edges near the corners (intersection of two or more edge segments). If no end points or line segments are present within its neighborhood, it is considered to be a vertex (connected to the other end of the line segment). However, if some end points or line segments are found in its neighborhood, a computation is performed to locate a best-fit vertex.

Let there be M line segments $[(\theta_i,\rho_i)$, $i = 1,M]$ which form the vertex. This list includes both the line segments found in the neighborhood and the line segments belonging to the end points found in the neighborhood. An example of this is shown in Figure 15. In this example, there is one end point P_2 and one line segment ℓ_3 present in the circular neighborhood of the end point P_1. Thus, there are a total of three line segments forming the vertex. The criterion used in determining the best-fit vertex is to minimize the total normal distance from the vertex to the line segments. Let the position of the best-fit vertex be denoted by (x_v,y_v). The normal distance from the vertex to a line (θ_i,ρ_i) is given by

$$d_i = \rho_i - x_v \cos \theta_i - y_v \sin \theta_i \ .$$

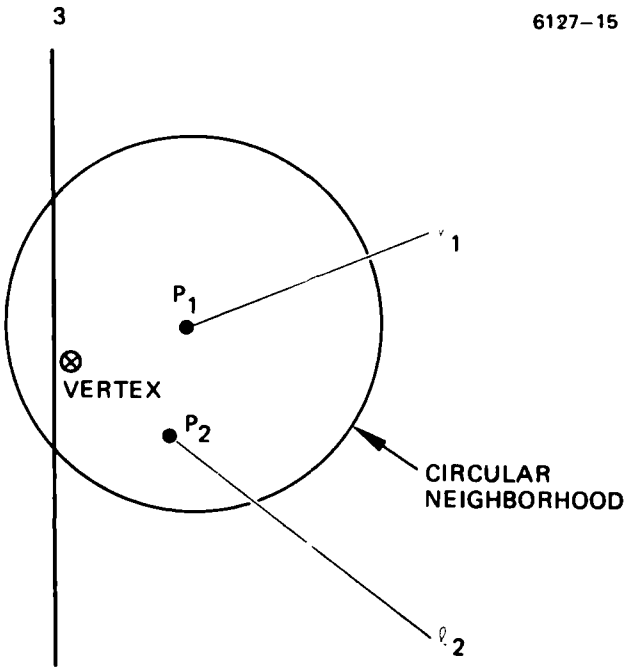


Figure 15. Locating end points and line segments in a circular neighborhood for a best-fit vertex computation.

The total normal distance D for the M line segments is given by

$$D^2 = \sum_{i=1}^{M} (\rho_i - x_v \cos \theta_i - y_v \sin \theta_i)^2 .$$

The x and y coordinates of the vertex are determined by solving the following two simultaneous equations which minimize D^2

$$\frac{\partial D^2}{\partial x_v} = 0$$

$$\frac{\partial D^2}{\partial y_v} = 0 .$$

A solution to these equations is presented in Appendix B. Once a vertex is located, the end points on the other side of the line segments which form the vertex are noted. This is done to derive the interconnections (links) between the vertex located and the other vertices already located or to be located.

These steps for locating a vertex and its connections are repeated for all the end points of the line segments. The vertex model is stored as a data structure consisting of the vertex label or number, its x and y coordinates, degree (number of other vertices that it is connected to), and the vertex numbers that it is connected to. Figure 16 shows the edge elements, line segments, and the vertex model for the block scene being considered. Table 4 gives the vertex positions and their links as found in this example.

E. VERTEX MODEL RESULTS ON OUTDOOR SCENES

Extracting straight-line edge segments and the subsequent vertex formation is more difficult for outdoor scenes. Difficulties arise mainly because of the presence (and detection) of too many edge elements and line segments in outdoor scenes. Some preliminary results are presented here for outdoor scenes containing house structures with background such as trees and bushes.

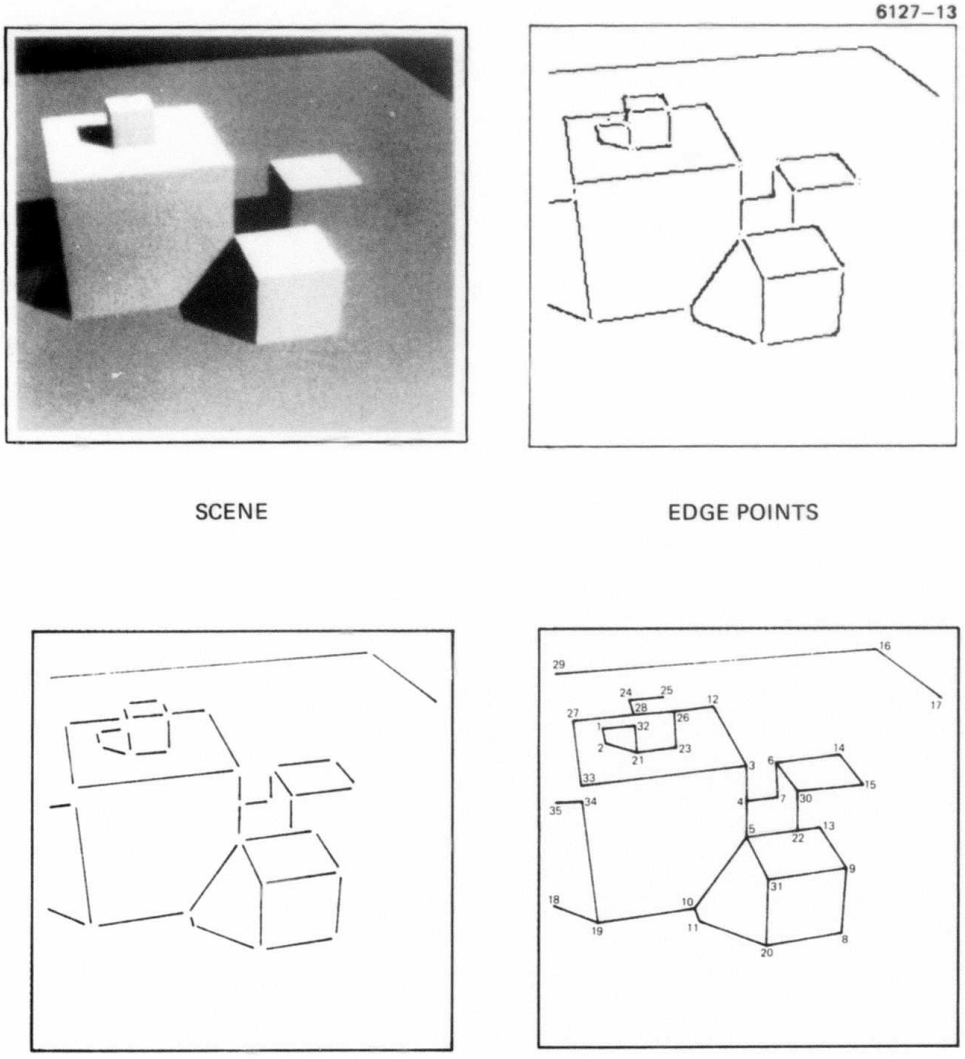


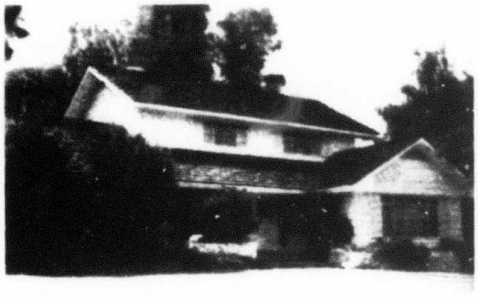
Figure 16. Edge points, line segments, and vertex model for a scene containing some block structures.

Table 4. Vertex Model for the Block Scene of Figure 16

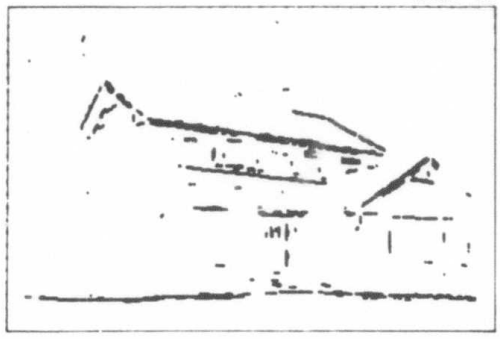
Vertex No.	x-y Position	Degree	Links (Connections)
1	(35 , 59)	2	2 32
2	(37 , 68)	2	1 21
3	(127 , 83)	3	4 33 12
4	(127 , 105)	3	3 5 7
5	(127 , 128)	4	4 22 31 10
6	(146 , 81)	3	7 30 14
7	(147 , 103)	2	6 4
8	(188 , 189)	2	9 20
9	(191 , 148)	3	8 31 13
10	(94 , 173)	3	11 5 19
11	(97 , 181)	2	10 20
12	(106 , 45)	2	3 26
13	(174 , 122)	2	9 22
14	(187 , 76)	2	15 6
15	(202 , 95)	2	14 30
16	(210 , 8)	2	17 29
17	(252 , 40)	1	16
18	(4 , 171)	1	19
19	(32 , 182)	3	18 34 10
20	(140 , 197)	3	11 31 8
21	(57 , 74)	3	2 32 23
22	(160 , 124)	3	5 30 13
23	(82 , 71)	2	21 26
24	(52 , 41)	2	25 28
25	(74 , 39)	1	24
26	(81 , 48)	3	12 23 28
27	(16 , 54)	2	28 33
28	(55 , 50)	3	27 24 26
29	(5 , 24)	1	16
30	(160 , 99)	3	22 6 15
31	(141 , 155)	3	20 5 9
32	(56 , 57)	2	21 1
33	(21 , 95)	2	27 3
34	(22 , 105)	2	19 35
35	(5 , 106)	1	34

Figure 17 shows the original scene, edge elements, line segments, and the resulting vertex model. Note that the edge elements shown are the ones obtained after the filtering operation. Some of the line segments are missed because of the extent of the edge filtering performed. However, most of the prominent line segments essential for a simple vertex model representation are found. The isolated short edge segments are not considered as part of the vertex model. This example shows that the vertex-formation step does not perform as well for near-parallel and close line segments. Some modifications are needed to consider such cases in a different way. Figures 18 and 19 also show similar results obtained on other house pictures.

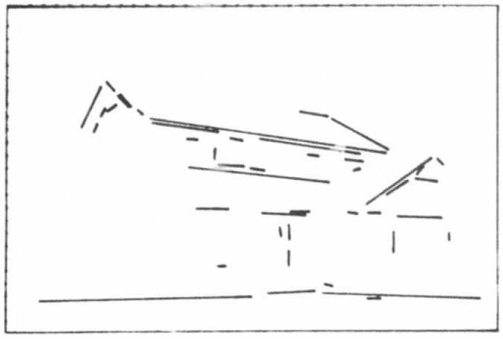
6127–23



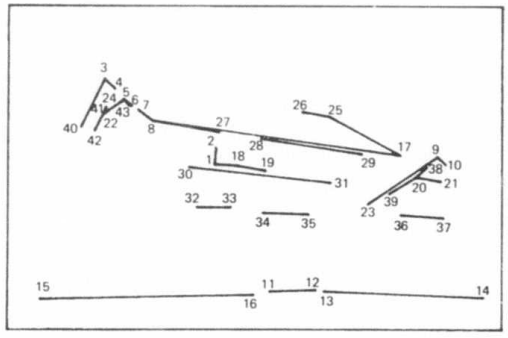
SCENE



EDGE ELEMENTS



LINE SEGMENTS

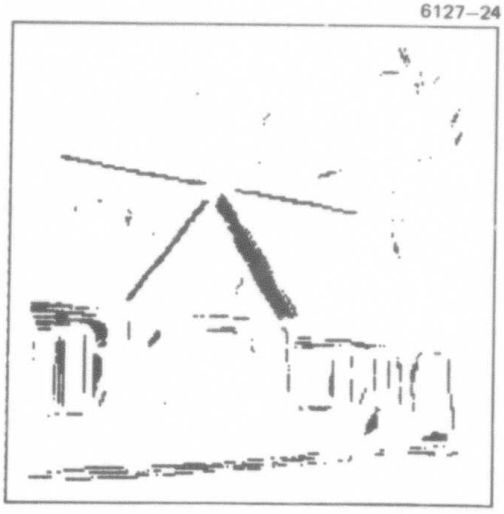


VERTEX MODEL

Figure 17. Vertex model for a scene containing a house structure with background of bushes and trees.

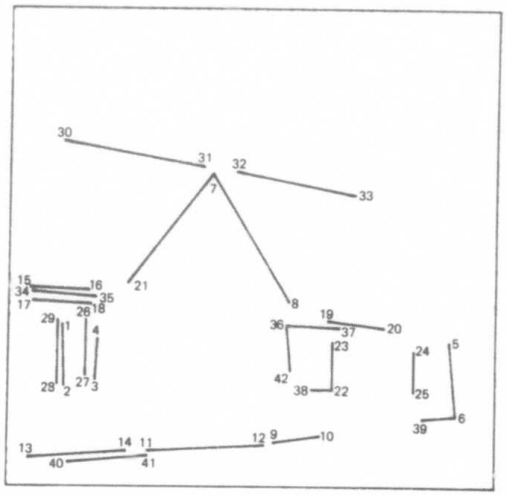
Figure 18. Vertex model for a scene containing a house structure with background of bushes and trees.

SCENE



EDGE ELEMENTS

LINE SEGMENTS



VERTEX MODEL

Figure 19. Vertex model for a scene containing a house structure with background of bushes and trees.

SECTION 5

SCENE MATCHING USING VERTEX MODELS

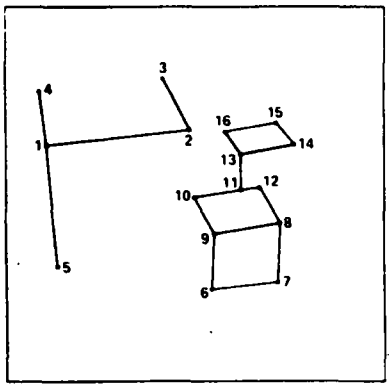
The procedure for constructing scene vertex models was described in the last section. The vertex model matching techniques are based on the information contained in the local structure of scene vertices. A local structure may be specified in terms of certain parameters which are determined for each scene vertex and stored as part of the model.[6] The following parameters are used:

- Position – x and y coordinates of the vertex
- Degree – number of lines meeting at the vertex
- Connections – labels of other vertices connecting to the vertex
- Subtending angles – angles subtended by adjacent links (counterclockwise) originating from the vertex
- Length ratios – ratios of the two links (counterclockwise) corresponding to each subtending angle at the vertex.

Since only one line originates from a vertex of degree 1, the subtending angles and the length ratios are not defined for that case. The vertices in the model are stored in order of decreasing degree, as a vertex of higher degree contains a greater amount of structural information. A simple example of a vertex model and the corresponding data structure is shown in Figure 20.

The matching technique compares the reference and sensed models to derive the registration information between the reference and the sensed scene. The reference vertex model is derived ahead of time from the given reference imagery. When building the reference model, only the most prominent and significant lines and the resulting vertices are chosen. This is done to ensure that most of the reference model will be visible and easily extractable from the sensed scene. Since the sensed model is determined on-line in most applications, there is no opportunity to select the most prominent features ahead of time.

6127–19



(a) VERTEX MODEL

VERTEX NO	x y	DEGREE	CONNECTIONS	ANGLES (DEGREES)	LENGTH RATIOS
8	189 149	3	12 9 7	69 78 214	0.6, 1.1 1.4
11	161 126	3	12 13 10	81 99 180	0.5, 0.7, 2.5
1	23 94	3	5 2 4	90 92 178	0.8, 2.7, 0.4
13	161 101	3	11 14 16	100 114 146	0.7, 2.0, 0.8
9	142 157	3	6 8 10	103 109 149	0.8, 1.6, 0.8
14	199 94	2	15 13	60 300	0.5, 1.9
16	150 85	2	13 15	65, 295	0.5, 1.9
2	125 84	2	3 1	68 292	0.4, 2.5
10	128 131	2	9 11	70 290	0.9, 1.1
6	140 196	2	7 9	81 279	1.2, 0.8
7	187 191	2	8 6	99 261	0.9, 1.1
12	174 124	2	11 8	112 248	0.5, 2.2
15	186 79	2	16 14	122 238	1.8, 0.5
4	18 56	1			–
5	31 180	1			–
3	106 47	1			–

(b) DATA STRUCTURE

Figure 20. Data structure used for a vertex model matching scheme.

Consequently, a larger number of lines and vertices are generally present in a sensed model than in the corresponding reference model. The matching procedure used in this scheme consists of two major steps – local and global vertex matching.

A. LOCAL VERTEX MATCHING

Two vertices, one from the reference and the other from the sensed model, are considered to have a local match if all the following criteria are met:

- Degrees of both reference and sensed vertices, D_r and D_t, respectively, are greater than 1
- $|D_r - D_t| \leq 1$
- D angular matches if $D_r = D_t = D$

  or

  D - 1 angular matches if $|D_r - D_t| = 1$ and $D = \min (D_r, D_t)$.

An angular match is said to be present if a subtending angle of the reference vertex is equal to that of a sensed vertex within a given tolerance, and also if their corresponding length ratios are equal within a given tolerance.

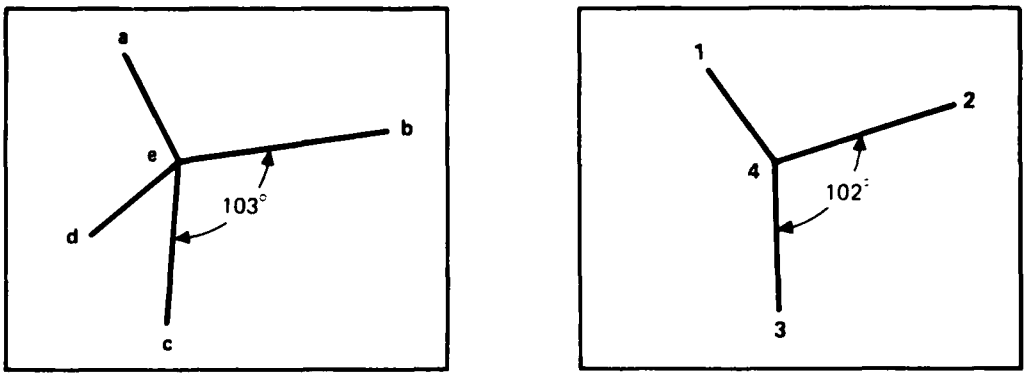
An example of a local match between two vertices is shown in Figure 21. The pairs of resulting matching vertices for this example are (4,e), (2,b), (1,a), and (3,c).

The vertices in a scene model are stored in order of their decreasing degree. The first vertex from the reference model is compared with vertices in the sensed model for a local match. If the degree of the reference vertex is D_r, the sensed vertices are considered for comparison in order of D_r, $D_r + 1$, and $D_r - 1$, for a local match. The $D_r + 1$ degree is considered before $D_r - 1$ since it is likely that a weak line, not considered as part of the reference model, may have appeared in the sensed model. This process is repeated for all the reference vertices to locate possible local matches with sensed vertices. Thus, a list of pairs of matching vertices is obtained through this step.

B. GLOBAL VERTEX MATCHING

Let the coordinates of the reference and sensed vertices, which were matched in the previous step, be specified by (x_k, y_k) and

6127-18



DEGREE = 4
ANGLES = 43°, 103°, 104°, 109°
LENGTH RATIOS = 0.71, 0.74, 1.63, 1.10

DEGREE = 3
ANGLES = 102°, 108°, 150°
LENGTH RATIOS = 0.82, 1.6, 0.76

ANGULAR MATCHES ⇨ 2-4-1 : b-e-a
3-4-2 : c-e-b

Figure 21. An example of a local match.

(u_k, v_k), respectively. A linear transformation between the reference and the sensed coordinate systems may be specified by the following equations:

$$u = ax + by + c$$

$$v = -bx + ay + d \ ,$$

where the parameters (a,b,c,d) define the scale, rotation, and translations between the two systems of coordinates. The set of vertices (x_k, y_k) and (u_k, v_k) is used to estimate the parameters (a,b,c,d) through the use of the following linear system of equations.[7]

$$\begin{bmatrix} \Sigma(x_k^2 + y_k^2) & 0 & \Sigma x_k & \Sigma y_k \\ 0 & \Sigma(x_k^2 + y_k^2) & \Sigma y_k & -\Sigma x_k \\ \Sigma x_k & \Sigma y_k & n & 0 \\ \Sigma y_k & \Sigma x_k & 0 & n \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \Sigma(u_k x_k + v_k y_k) \\ -\Sigma(v_k u_k - u_k y_k) \\ \Sigma u_k \\ \Sigma v_k \end{bmatrix}$$

The reference vertices for which no match has been found through the local matching step are now mapped to the sensed coordinate system using the estimated values of parameters (a,b,c,d). Each mapped reference vertex is matched to its nearest neighbor, within a specified tolerance, from the sensed vertex list. A reference vertex is said to have no match if there is no neighbor found for its mapped position in sensed coordinates within the specified tolerance. This global matching step will not be necessary in situations when suitable matches have been found during the local matching step for all the given reference vertices.

The pairs of matching vertices between the reference and sensed scenes found during the local and global matching steps are called the control points. These control points are then used to obtain the proper registration between the reference and sensed scenes in one of two ways: (1) digital linear warp to force correspondence of the control points in the two scenes and (2) estimating the aspect position

of the platform (sensor) with respect to the area of interest. Through either of the above registration methods, one can determine the position of a particular point of interest (known in the reference coordinates) in the sensed scene.

The vertex model matching approach has been tested on several scenes. Figure 22 shows the reference and sensed scenes and their corresponding models. The reference model, which was purposely made simple, consists of only 12 vertices. However, the sensed model is more complex and contains many more vertices than the reference model. A portion of the reference imagery is not present in the sensed imagery. Also note that the two images are not related by a simple linear transformation. The matching results on these two scenes are given in Table 5. This table gives the reference vertices and their corresponding match from the sensed vertices. There were no matches found for the reference vertices a and d, as they are outside the field of view in the sensed imagery.

An example of scene matching on a low resolution complex visible imagery of a building will now be considered. Figure 23 shows the steps of vertex model construction on this visible imagery. The image resolution in this example is relatively poor. Several isolated and short line segments were filtered out to obtain a simpler vertex model. The reference model in this example was constructed by hand, approximately to scale with its corresponding structures in the visible imagery. As the reference model in Figure 24 shows, only some of the prominent features (such as the cylindrical structure and the windows) are considered part of the reference model. For most of the reference vertices, correct matches are found in the sensed model. No matches are found for reference vertices 7 and 8 because they are outside the field of view in the sensed imagery.

The vertex model matching was also performed on an infrared imagery of the same building scene. Figure 25 illustrates the steps involved in vertex model construction. The original imagery was first low-pass filtered (averaging) to reduce degradation from the presence of speckle in the imagery. The matching results, similar to those

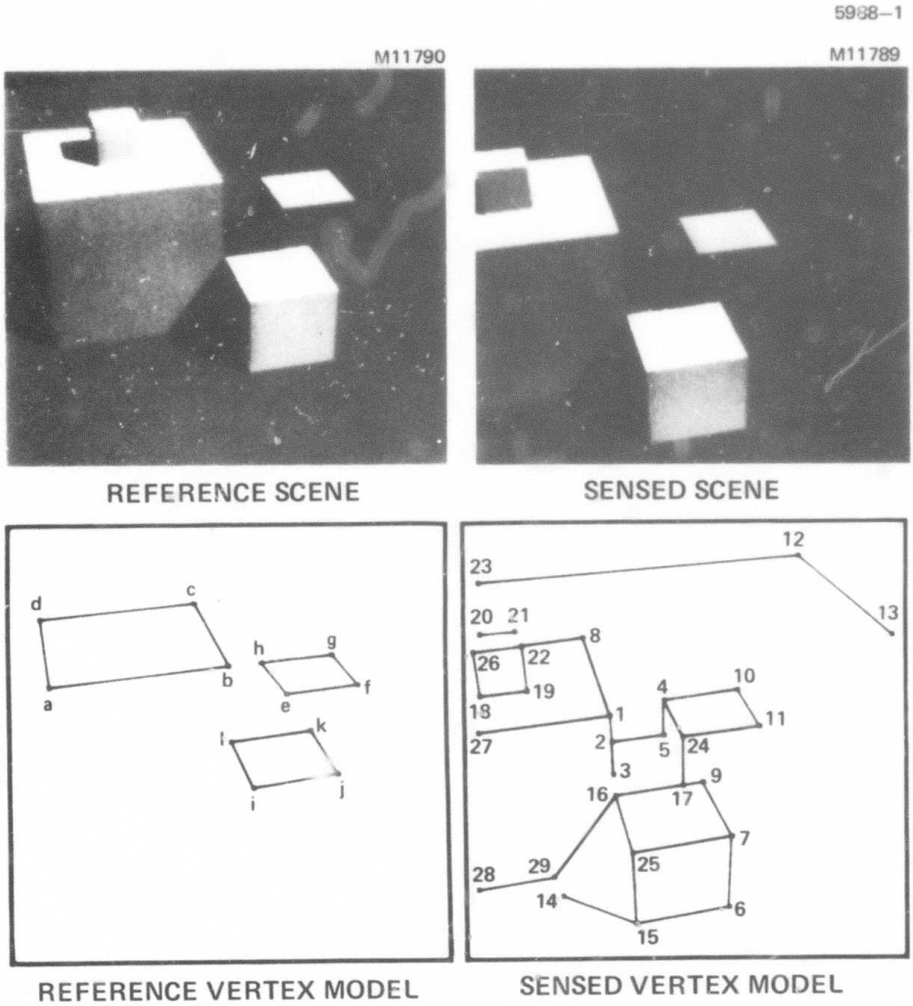


Figure 22. The reference and sensed scenes with their corresponding vertex models.

Table 5. Vertex Model Matching Results on Reference and Sensed Scenes of Figure 22

6127–20

MATCHING VERTICES			
REFERENCE		TEST	
NODE	(X, Y)	NODE	(U, V)
a	(20, 94)	NOT FOUND	
b	(126, 82)	1	(84, 114)
c	(106, 44)	8	(68, 67)
d	(15, 53)	NOT FOUND	
e	(160, 99)	24	(127, 127)
f	(202, 94)	11	(173, 121)
g	(187, 76)	10	(160, 99)
h	(145, 80)	4	(116, 105)
i	(140, 155)	25	(97, 197)
j	(190, 147)	7	(156, 187)
k	(174, 121)	9	(139, 154)
l	(127, 127)	16	(87, 162)

6057–4

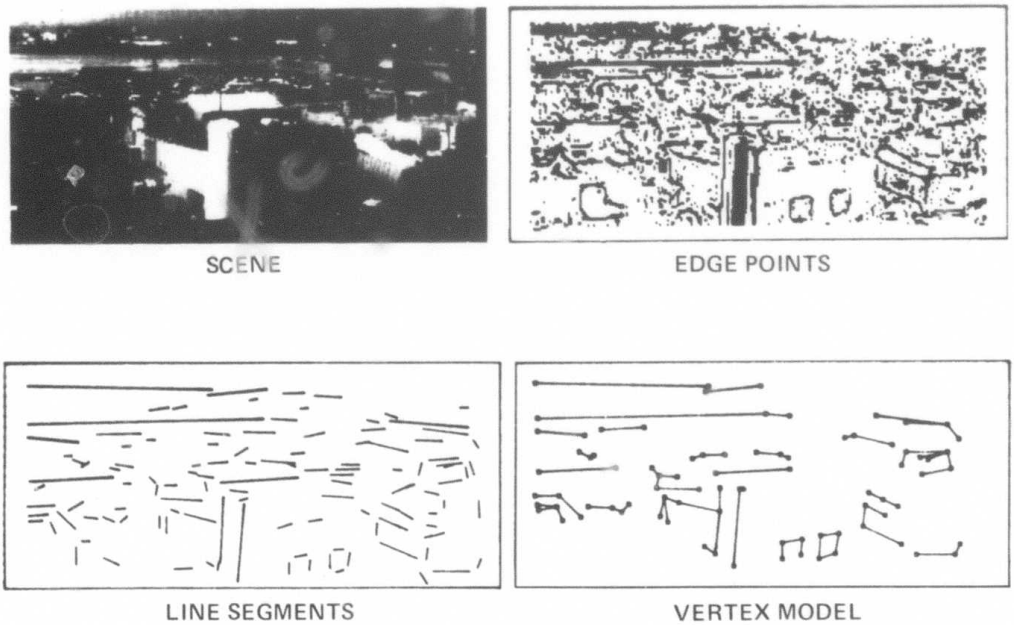
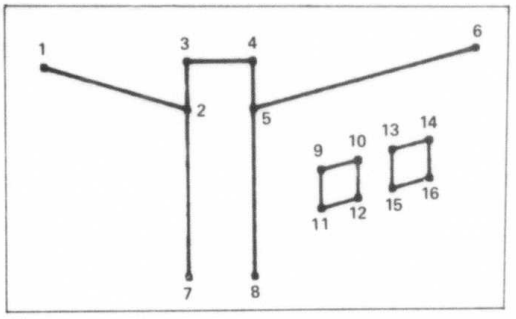
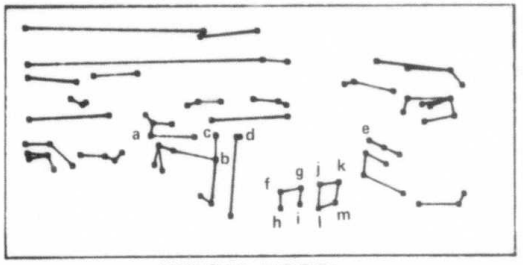


Figure 23. Vertex model construction from a low resolution visible image of a building.

6057–13



REFERENCE MODEL



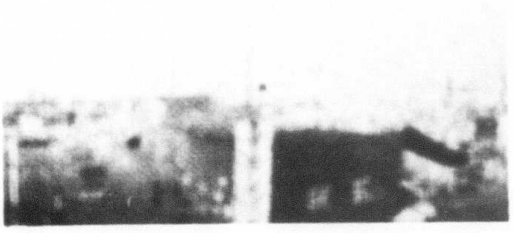
SENSED MODEL

MATCHING VERTICES

REFERENCE	SENSED
1	a
2	b
3	c
4	d
5	NOT FOUND
6	e
7	NOT FOUND
8	NOT FOUND
9	f
10	g
11	h
12	i
13	j
14	k
15	l
16	m

Figure 24. Vertex model matching results on visible image of Figure 23.

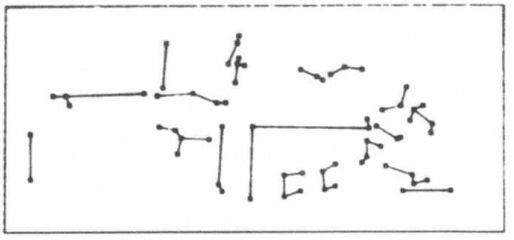
6057–3



LOW-PASS FILTERED SCENE

EDGE POINTS

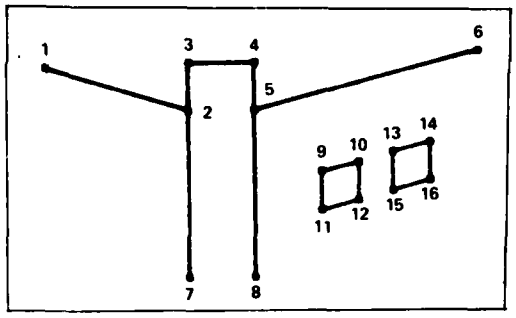
LINE SEGMENTS



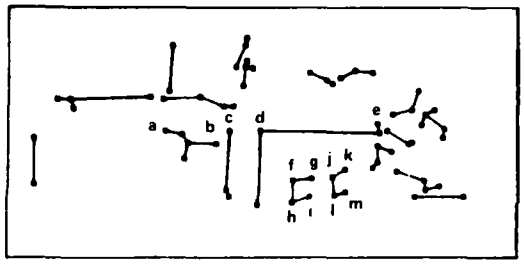
VERTEX MODEL

Figure 25. Vertex model construction from an infrared image of a building.

obtained for visible imagery, are shown in Figure 26. The pairs of matching vertices (control points) between the reference and the sensed models are sufficient to derive the registration information between the two scenes.



REFERENCE MODEL



SENSED MODEL

6057–12

MATCHING VERTICES

REFERENCE	SENSED
1	a
2	b
3	c
4	d
5	NOT FOUND
6	e
7	NOT FOUND
8	NOT FOUND
9	f
10	g
11	h
12	i
13	j
14	k
15	l
16	m

Figure 26. Vertex model matching results on infrared image of Figure 25.

SECTION 6

SCENE REPRESENTATION AND MATCHING USING REGION MODELS*

A complementary approach to vertex-based models is based on extracting and using regions (areas) in an image that correspond to meaningful entities, such as surfaces of a single object or a group of objects as is typical of suburban areas. For many objects, such regions are uniform in some image property (e.g., brightness, texture, or "color"). Objects such as bodies of water are expected to have uniform (or slowly varying) brightness; mountainous or suburban areas may have large sections that have a uniform texture. Although the region-based models are most appropriate for long-range scenes, one can use them to obtain positions of points of interest which are small and unresolvable. This can be done by defining their positions with respect to some large key regions.

A model of the scene can be built from the regions by describing their properties, such as size and shape, and the relationships of the various regions to each other. The properties of a single region may be adequate for its recognition, if it has a distinguishing shape. However, groups of regions will need to be examined as a unit to resolve similar looking regions.

There are two major problems with this approach: how to reliably extract the desired regions (the segmentation problem) and how to correctly identify the regions in the presence of errors caused by noise or the segmentation procedures. A simple scheme to extract areas of uniform brightness and some results are presented in this section. The use of such regions for matching and associated problems are also discussed here.

A. REGION EXTRACTION

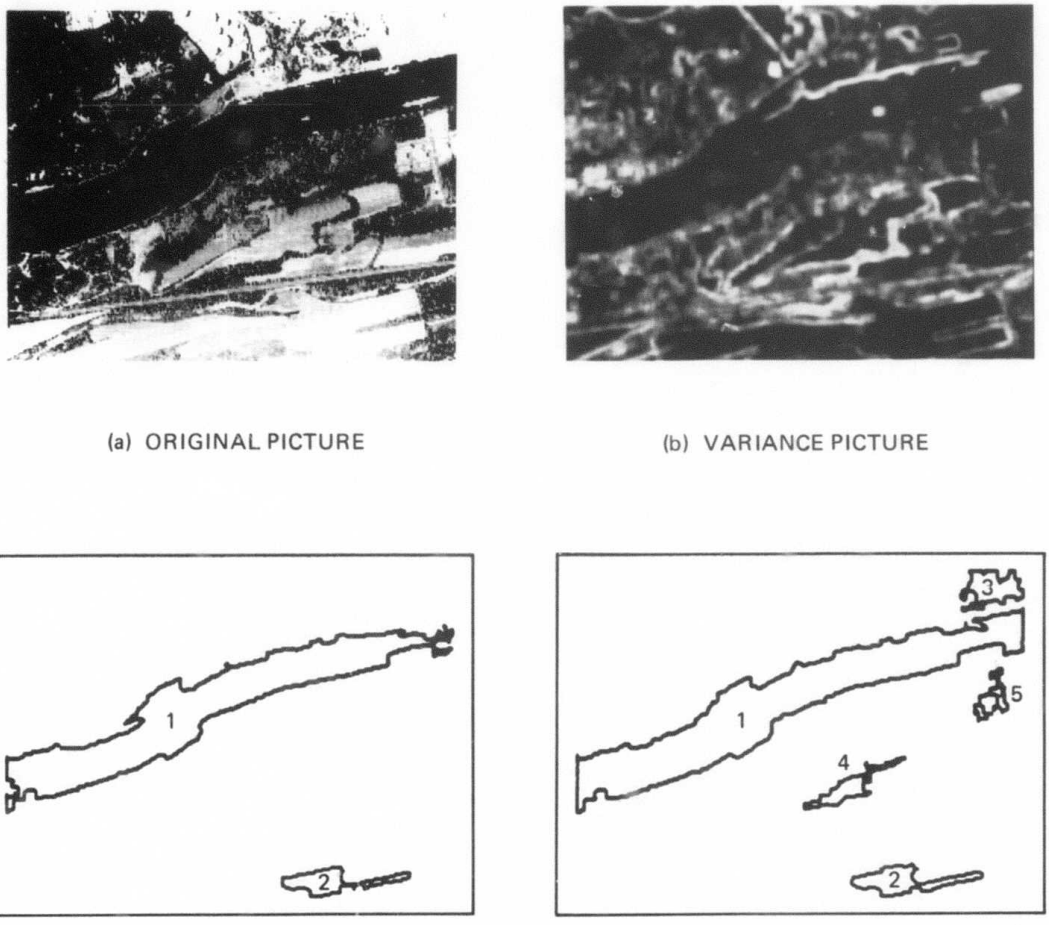
Several approaches to scene segmentation have been explored by various researchers. One approach involves edge detection and linking of these edges to give region boundaries. Since edge detection is

*Professor Ramakant Nevatia of the University of Southern California provided the initial draft for this section.

local in character, this approach often fails to yield closed regions. Also, it is difficult with this approach to detect edges in presence of texture and to follow irregular region boundaries. Another approach is to start with small regions of a constant property and merge these into larger regions by a chosen selection procedure.[8] The success of such techniques for general, natural scenes has been limited. A third approach is to segment a complete image into successively smaller regions.[9,10] Histograms of various attributes of a region are examined. If an attribute displays a marked peak or a bimodal distribution then the region is split into smaller regions corresponding to the different parts of the histogram distribution. The success of this technique is dependent first on the proper identification of the desired distributions in the histograms and second on the proper choice of image attributes. This technique has been successfully demonstrated on some natural, color pictures, but little work has been done with black and white images.

To avoid some of these difficulties, we have developed a simple, straightforward technique for extracting regions of uniform brightness by computing the variance of image brightness in a local area and choosing the areas of low variance. This method does not require the region property to be constant, but only that the variations be smooth and small. A square window is centered around each pixel of the image (except near the picture borders, which are ignored). Variance of the brightness in this window is computed and then assigned to the center pixel, thus producing a new matrix of numbers which may be viewed as a "variance picture." From this variance picture we extract regions which are composed of values below a certain selected threshold. We use an algorithm developed by Dudani[11] for extracting regions using boundary following. This procedure gives several separate contiguous regions, the boundaries of which are also available. The current implementation does not detect any holes in a region.

Results of processing two images using this technique are shown in Figures 27 and 28. Figures 27(a) and 28(a) show the two images. Figures 27(b) and 28(b) show the corresponding variance pictures



(a) ORIGINAL PICTURE

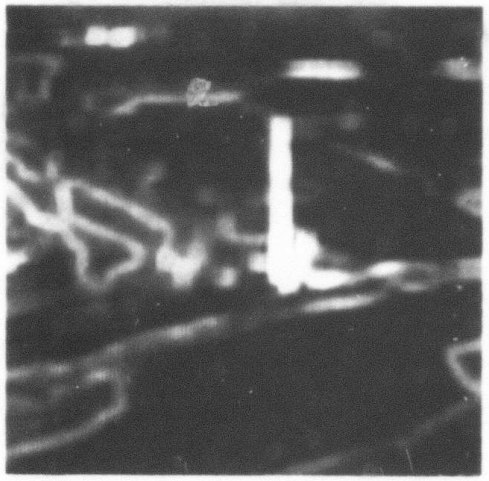
(b) VARIANCE PICTURE

(c) REGIONS WITH VARIANCE ⩽ 4

(d) REGIONS WITH VARIANCE ⩽ 6

Figure 27. Uniform intensity region extraction on a scene containing a river.

(a) ORIGINAL PICTURE



(b) VARIANCE PICTURE

(c) REGIONS WITH VARIANCE $\leqslant 4$

(d) REGIONS WITH VARIANCE $\leqslant 6$

Figure 28. Uniform intensity region extraction on a scene containing a lake and a power plant.

(lower variance points are displayed to be darker). A window of 5 x 5 pixels was used here. Figures 27(c) and 28(c) show the boundaries of regions extracted from pixels whose variance was less than or equal to the selected maximum of 4 (the original picture grey levels range from 0 to 255). Only regions large enough to contain at least 100 points in their boundaries have been retained. Figures 27(d) and 28(d) show the regions extracted by using a variance threshold of 6.

These results show that large key uniform areas, the large bodies of water in the examples, have been extracted successfully [region 1 in Figure 27(c) and region 4 in Figure 28(c)]. Extraction of one or more such key regions is the main objective for this simple technique. A comparison of Figures 27(c) with 27(d), and Figures 28(c) with 28(d) indicates the insensitivity of the operations to the choice of variance threshold. As points with higher variance are accepted, more regions are found, but the important key regions remain unchanged.

The major parameter of importance is the window size used for computing variances. The window size is chosen to be much smaller than the size of the expected key regions. In general, the region boundaries may be in error up to the size of the window unless post-processing to "grow" the extracted regions is performed. Note that the window needs to be large enough so that variations caused by local noise are not significant. A possible alternative method for optimizing window size is to use windows of different size and to combine the results. In principle, our technique could be applied to extract regions that are uniform in some property other than brightness. Texture features might be useful for analyzing natural scenes. Use of simple texture features, such as coarseness, will extend the types of regions that can be extracted. However, texture analysis is a major problem of image analysis and the complex texture features, such as those described in Ref.12, are expensive to compute and not necessarily always adequate.

B. MATCHING OF REGION MODELS

To identify regions in a scene requires matching against a previously computed or hand-stored model of the scene. Such matching may be viewed as having two major parts. One is the matching of isolated regions based on their properties (such as shape). The other is to check for the consistencies of the relationships of regions to each other to reduce the ambiguities of the first step. In this, the region model matching problem is similar to that of vertex model matching, but with different properties and different relationships, and the same techniques may apply. This homogeneity will also allow region-based and vertex-based models to be easily integrated. An important property is the shape of a region. If regions extracted from two images are consistent and differ only in scale, translation, and rotation, then simple geometric shape measures may suffice. Many shape measures for simple figures are available in the literature. The development of region properties useful for region models in our work is in a preliminary stage. Some of the measures that we are considering are

- The area of the region (number of pixels in the region)
- The perimeter (also in pixel units)
- 4π* area/perimeter^2
- The radius of gyration, $\gamma = (\mu_{20} + \mu_{02})^{1/2}$
- A second moment measure, defined as

  $$\frac{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}{\gamma^4}$$

  where μ_{pr} stands for pr^{th} central moment (see Ref.13 for details).

The third measure determines the closeness of a region to a circle and the fifth one is related to the "elongation" of a region. Table 6 shows the computed values for these five measures for the regions of Figures 27(c) and 28(c). The examples shown here are not sufficient

Table 6. Shape Measures for Regions in Figures 27(c) and 28(c)

Region Labels	Shape Measures				
	1	2	3	4	5
Figure 27(c)					
1	980.8	5722.0	13.3	95.2	0.24
2	241.9	493.0	9.4	21.6	0.83
Figure 28(c)					
1	336.1	297.0	30.1	35.7	0.58
2	196.1	424.0	7.2	36.2	0.95
3	150.9	355.0	5.1	27.6	0.95
4	419.3	2788.0	5.0	40.6	0.68

to evaluate the comparative usefulness of these measures; a simple area measure is adequate for the examples shown.

Shape matching for isolated regions is difficult primarily because of variations caused by small changes in the scene. For example, the region in Figure 27 corresponding to the river will break into two or more regions if a small portion across its width is obscured. To handle such variations, the matching process will need to consider alternatives of merging or splitting of different regions. Segmentation into simpler shapes may help with the obscuration of parts of a region.[14]

A region model will also contain the relationships between the different regions. Simple relationships between regions are the metric ones and consist of the relative positions of the various regions. These relationships can be used in the same way as for the vertex-based model matching.

SECTION 7

CONCLUSIONS AND FUTURE WORK

Encouraging results from our work on model-based scene-matching schemes lead us to believe that our approach is sufficiently sound and robust to perform adequately with complex real imagery appropriate to various possible mission scenarios. Some of these results were given in this report; in particular, the feasibility demonstration of the vertex-based model-matching system for registering two scenes. Although much additional work is needed to develop our image-processing techniques into a fully automatic, workable image-based guidance system, we are confident of success. Many parts of this system have, in fact, already been perfected; interfacing and feedback loops between the different parts do, however, remain to be developed. The needed modifications are outlined in this section.

Experimental results on scene matching using vertex models were reported in section 5. These results illustrate the robustness and inherent power of model-based matching techniques. The approach has the advantage that the reference model is simple and has low data requirements. The reference model is constructed before it is needed; it is often only necessary to store a list of points (vertices) and their interconnections, and it is feasible to store multiple references. Also, any modifications or changes needed to update the reference model can be conveniently specified. The scale of the reference model shown in Figure 24 is approximately proportional to the scale of the corresponding structure present in the visible image. This also illustrates the robustness of the matching technique, since the model was drawn with little attention paid to the exact positions of the vertices. In addition to the inexact scale of the reference model, certain portions of structures included in the reference were outside the field of view of the sensed scene. Correct matching results were obtained in spite of these differences because there was enough useful structural information present in the reference and sensed models. The same reference

model was used successfully to match both the visible and the infrared imagery, demonstrating that the technique is insensitive to which of these sensor types is used. Another strong point in our approach is that the results are presented in terms of matched control points between the reference and sensed scenes. This allows our system to tolerate a large degree of three-dimensional distortion between the reference and sensed scenes. As pointed out in Section 5, these control points can be used to determine the exact three-dimensional transformation between the two views.

Our approach also has low computational requirements. In a software implementation on a general-purpose machine of the model construction and matching techniques, the majority of computing required was at the lowest level (e.g., edge detection). Edges are currently detected by the Hueckel operator, which is a complex, time-consuming process. However, the Hueckel operator apparently can be replaced with a simpler edge-detection technique. This technique can be implemented in hardware for real-time operation. In fact, at HRL we are investigating the possibility of developing chips to perform the entire low-level edge-detection process. The time required for the other model-generation steps is relatively small, and we expect it to be within the requirements for most practical scene-matching applications. The extremely small computing time necessary for the final step in matching the reference and sensed models is quite significant since it provides a margin permissive to adopting modifications that improve the algorithm's sophistication (in the sense of taking advantage of more of the available information) but which require longer computing times. The techniques used in our approach are believed to be implementable in a final working system without any major difficulties in hardware or software.

Substantial development of the feature-extraction algorithm is needed before region models can be used for matching two scenes. The main difficulty, as pointed out in Section 6, lies in the region segmentation techniques necessary to locate the key regions; these in turn provide the necessary features for the model. Once we can successfully

construct region models, the rest is quite simple since the matching scheme will be similar to that employed in vertex models.

Most scene-analysis techniques have some internal parameters, the values of which are preassigned. Generally speaking, a technique should be designed such that the resulting performance is totally independent of the exact values of the internal parameters. However, it is sometimes advantageous to set the parameters each time based on the requirements of the particular imagery. This additional capability can be provided through a goal-directed executive control program. Such a program is shown in Figure 29; it has several communication links between the control program and different portions of the system (as presently implemented). These links and their respective roles are discussed below.

Most edge-detection techniques use some form of a threshold parameter, the effect of which is to limit the number of edge elements. If this parameter is set such that only very strong edges are obtained, some important, but weak, edges are usually missed. However, setting the parameter so as to also obtain the weak edges results in an enormous number of edges being detected, including many useless noise-generated

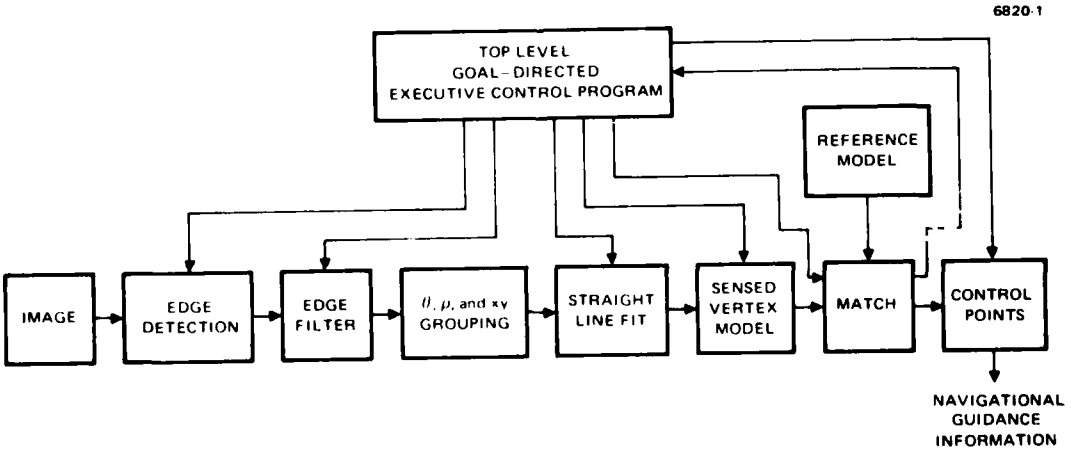


Figure 29. Block diagram of a complete image-based guidance system controlled by a goal-directed executive control program.

edges. One way to solve this problem is to first obtain only the strong edges, and then, as necessary, look for weak edges selectively in certain designated areas. This would require some form of communication between the edge-detection and the matching steps. In Figure 29, this is provided through the control program.

The edge-filtering process also has two internal parameters, n and ℓ, as described in Section 4.B. A strategy similar to one described in the last paragraph for edge detection can also be used for this step, with the values of n and ℓ set through the goal-directed control program.

There is no communication link shown in Figure 29 between the executive control program and the steps of the θ, ρ, and xy grouping techniques (described in Section 4.C.). Indeed, there are several internal parameters for these steps, but in these cases their values can be preset and need not be changed. The input to these steps are the filtered edge elements, from which several groups are formed for fitting straight lines. Since the processing in these steps does not depend on any particular goal, their internal parameters can be preset.

As described in Section 4.C.5, there are two measures of goodness of fit of a straight line, $\overline{d}$ and η. These two measures can be used to select the line segments for constructing the vertex model. For example, a value of zero for $\overline{d}$ indicates an ideal fit; with an appropriate upper bound on $\overline{d}$, only the most prominent line segments can be selected. Also, the line segments can be chosen on the basis of their length. The acceptable values for these measures may be provided through a link from the executive control program.

The vertex model is constructed from the line segments extracted. In this part of the processing, two parameters must be specified: a distance threshold below which vertices are merged into one and the minimum length requirement for the inclusion of isolated line segments in the vertex model. These parameters may also be provided through the executive control program.

A key step in the complete system is the matching of the reference and the sensed models to obtain the control points. The acceptable matching angle and length-ratio tolerance (described in Section 5.A),

are specified through this control program. Initially, matches with low tolerances can be obtained; then later, if necessary, the tolerances can be increased to increase the number of possible matches. Also, this step can direct (through the executive control program) other parts of the system to repeat certain processing steps with the adjustable parameters changed. For example, if a line segment is present in a certain part of the image but has not yet been located, then the edge-detection step can be made to look for weak edges in that part of the image.

Finally, once a list of control points is obtained by matching the reference and the sensed models, the necessary guidance and aimpoint information can be extracted from this list. These results are provided through the goal-directed executive control program.

In addition to developing the hierarchical, structural program discussed above, the following areas of the model-based scene-matching system were identified as needing further work:

- Simple edge-detection techniques to replace the Hueckel operator need to be developed and tested, and their effects on line-segment extraction determined.

- A modified filtering technique that keeps only the edges belonging to consecutive line segments with radii of curvative within specified ranges needs to be implemented. Such a technique could be used to extract edge elements belonging to curvilinear objects in a scene.

- The θ and ρ grouping steps seem to function very well. However, the xy grouping step could be improved by deriving more information from the subgroups of edge elements under consideration. In our present implementation, the distance between two edge points is the only condition tested for merging purposes. Although two elements from two different xy groups lie close together, the majority of the respective group edge point constituents may be far apart, indicating that the two groups occupy different local areas in the scene. Therefore, the mean and standard deviation of the edge positions of each xy group should be computed and used as additional merging criteria. In this way, bodies of edge-element groups that are far apart will not be merged even if some of their neighboring edge points lie close to one another, and each subgroup will be subjected to a straight-line fitting process.

- Some modification is needed in our present vertex model construction technique for handling cases of parallel line segments lying close to each other.

- Some important modifications are needed in our model matching techniques. As described in Section 5, the vertices are matched by using their local structures. An additional step should be introduced following this local matching. After a local match is obtained, it should then be checked for connective and positional consistency. Connective consistency confirms that matching vertices are connected to vertices that also match. Similarly, positional consistency confirms that matching vertices are connected to other vertices having approximately the same relative positions. A measure for the match of two vertices would also be useful. Such a measure could be used to structure the matching technique so that it gives greater importance to matched vertices with high measures.

In conclusion, we feel that our techniques for model-based scene matching will, when perfected, yield a fully automatic, workable image-based guidance system for several important applications.

APPENDIX A

FITTING A STRAIGHT LINE TO N GIVEN POINTS

This appendix derives the equations used for determining a best-fit straight line to N given points. The criterion used for determining such a straight line is to minimize the total normal distance from given points to the straight line.

Let the N given points be $\{(x_i, y_i),\ i = 1, N\}$ and let the straight line be represented by the following relation:

$$\rho = x \cos \theta + y \sin \theta \ . \tag{1}$$

The normal distance of a point (x_i, y_i) to the straight line (Figure A-1) is given by

$$d_i = \rho - x_i \cos \theta - y_i \sin \theta \ . \tag{2}$$

Let D be the total normal distance from the N given points to the straight line; it is given by

$$D^2 = \sum_{i=1}^{N} (\rho - x_i \cos \theta - y_i \sin \theta)^2$$

$$= N\rho^2 + \cos^2 \theta \, \Sigma x^2 + \sin^2 \theta \, \Sigma y^2 - 2\rho \cos \theta \, \Sigma x$$

$$- 2\rho \sin \theta \, \Sigma y + 2 \sin \theta \cos \theta \, \Sigma xy \ . \tag{3}$$

The parameters (θ, ρ) are obtained through solving the two simultaneous equations which minimize the square of the total distance

$$\frac{\partial D^2}{\partial \theta} = 0 \tag{4}$$

$$\frac{\partial D^2}{\partial \rho} = 0 \tag{5}$$

6127–15

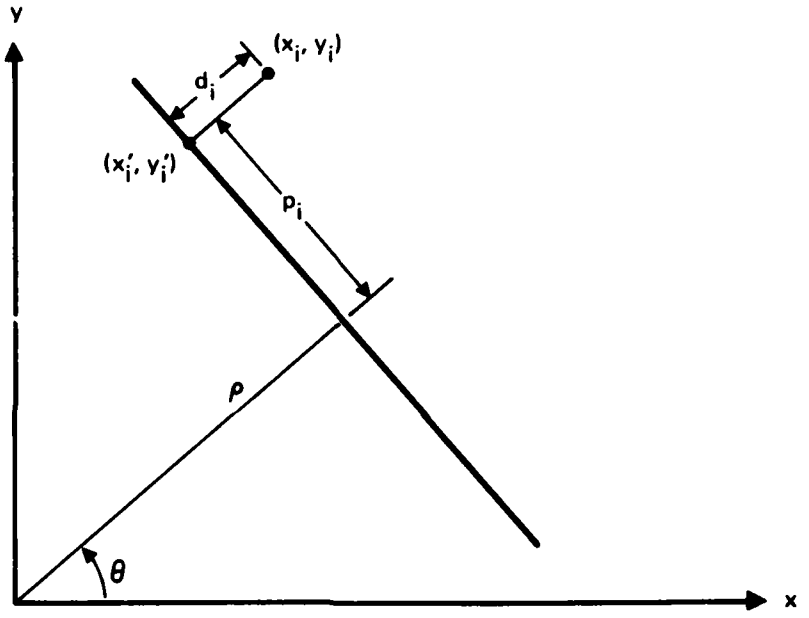


Figure A-1. Straight-line fit to N given points.

Substituting the value of D^2 into Eqs. 4 and 5 yields

$$\frac{\partial D^2}{\partial \theta} = 2\left\{\sin\theta\cos\theta\,[\Sigma y^2 - \Sigma x^2] + \rho\sin\theta\,\Sigma x - \rho\cos\theta\,\Sigma y + \Sigma xy\,[\cos^2\theta - \sin^2\theta]\right\} = 0 \qquad (6)$$

$$\frac{\partial D^2}{\partial \rho} = 2N\rho - 2\cos\theta\,\Sigma x - 2\sin\theta\,\Sigma y = 0. \qquad (7)$$

Thus, from Eq. 7 we have

$$\rho = \cos\theta\,\frac{\Sigma x}{N} + \sin\theta\,\frac{\Sigma y}{N}\ . \qquad (8)$$

Substituting ρ into Eq. 6 yields

$$A\sin\theta\cos\theta + B\cos^2\theta - B\sin^2\theta = 0, \qquad (9)$$

where $$A = \Sigma y^2 - \Sigma x^2 + \frac{(\Sigma x)^2 - (\Sigma y)^2}{N}$$

$$B = \Sigma xy - \frac{\Sigma x \Sigma y}{N}\ .$$

Eq. 9 is further simplified as shown below for two different cases based on whether or not $\sin\theta\cos\theta$ equals zero.

- Case 1: $\sin\theta\cos\theta = 0$ (i.e., either $\theta = 0°$ or $\theta = 90°$). In this, the value of $(\Sigma xy - \frac{1}{N}\Sigma x\Sigma y)$ is small (note that if this value is not below a certain small quantity, then (θ,ρ) is computed as discussed in Case 2). First compute the variances in x and y directions.

$$\sigma_x^2 = \frac{1}{N}\Sigma x^2 - (\frac{1}{N}\Sigma x)^2 \qquad (10)$$

$$\sigma_y^2 = \frac{1}{N}\Sigma y^2 - (\frac{1}{N}\Sigma y)^2\ . \qquad (11)$$

If $\sigma_x^2 > \sigma_y^2$, then $\theta = 90^o$; if $\sigma_y^2 > \sigma_x^2$, then $\theta = 0^o$. This is intuitively obvious since a horizontal line will have a very small variance in the y direction, but a relatively large variance in the x direction.

- Case 2: $\sin\theta \cos\theta \neq 0$ (i.e., $\Sigma xy - \frac{\Sigma x \Sigma y}{N} \neq 0$)

Dividing Eq. 9 by $\sin\theta \cos\theta$ and letting $C = \tan\theta$, we have

$$BC^2 - AC - B = 0 . \tag{12}$$

The two solutions for C from this quadratic equation are

$$C_1 = \tan\theta_1 = \frac{A + \sqrt{A^2 - 4B^2}}{2B} \tag{13}$$

$$C_2 = \tan\theta_2 = \frac{A - \sqrt{A^2 - 4B^2}}{2B} . \tag{14}$$

Substituting θ_1 and θ_2 into Eq. 8 gives the two values of ρ as follows:

$$\rho_1 = \cos\theta_1 \frac{\Sigma x}{N} + \sin\theta_1 \frac{\Sigma y}{N} \tag{15}$$

$$\rho_2 = \cos\theta_2 \frac{\Sigma x}{N} + \sin\theta_2 \frac{\Sigma y}{N} \tag{16}$$

Thus the two possible solutions are (θ_1,ρ_1) and (θ_2,ρ_2). The solution which gives the minimum distance as computed in Eq. 3 is chosen as the best-fit straight line to the given points.

The parameters θ and ρ give the position and direction of the best-fit straight line. At this point, one does not know the two end points of the line in x-y space. The two end points of the line can be determined from the given list of N points.

Each point (x_i,y_i) is projected on the best-fit line. The projection point (x_i',y_i') and the projection distance p_i as shown in Figure A-1 are given by

$$x_i' = x_i + \rho \cos\theta - x_i \cos^2\theta - y_i \sin\theta \cos\theta \qquad (17)$$

$$y_i' = y_i + \rho \sin\theta - x_i \cos\theta \sin\theta - y_i \sin^2\theta \qquad (18)$$

$$p_i = -x_i \sin\theta + y_i \cos\theta \; . \qquad (19)$$

The projection distances (p_i's) are sorted to find their minimum and maximum values. The (x_i',y_i') points corresponding to the two extreme projection distances are considered to be the two end points.

There are two measures of goodness of fit which may be used:

- The root-mean-square (rms) distance from a point to the line; this is given by

$$\bar{d} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\rho - x_i \cos\theta - y_i \sin\theta)^2} \; .$$

  A value of zero indicates an ideal fit.

- The number of points constituting the edge segment divided by the length of the line

$$\eta = \frac{N}{\ell c} \; ,$$

  where ℓ is the distance between the two end-points and c is a compensating factor based on the slope of the line:

$$c = \cos\theta \quad \text{if} \quad |\tan\theta| \leq 1$$

$$= \sin\theta \quad \text{if} \quad |\tan\theta| > 1 \; .$$

APPENDIX B

LOCATING A BEST-FIT VERTEX

This appendix presents the derivation of the equations used in determining the position of a best-fit vertex as described in Section 4.D. The best-fit vertex (x_v, y_v) is defined as the position minimizing the total normal distance from the vertex to the line segments forming the vertex.

The normal distance d_i from the vertex to the i^{th} line segment, represented by (θ_i, ρ_i), is given by:

$$d_i = \rho_i - x_v \cos \theta_i - y_v \sin \theta_i \ .$$

The total normal distance D for the M line segments forming the vertex is given by:

$$D^2 = \sum_{i=1}^{M} (\rho_i - x_v \cos \theta_i - y_v \sin \theta_i)^2$$

$$= \Sigma[\rho_i^2 + x_v^2 \cos^2 \theta_i + 2x_v y_v \cos \theta_i \sin \theta_i + y_v^2 \sin^2 \theta_i$$

$$- 2\rho_i x_v \cos \theta_i - 2\rho_i y_v \sin \theta_i] \ . \qquad (1)$$

The position (x_v, y_v) is determined by solving two simultaneous equations which minimize the distance D^2:

$$\frac{\partial D^2}{\partial x_v} = 0 \qquad (2)$$

$$\frac{\partial D^2}{\partial y_v} = 0 \ . \qquad (3)$$

Using the expression for D^2 in Eqs. 2 and 3, we have

$$\frac{\partial D^2}{\partial x_v} = \Sigma[2x_v \cos^2 \theta_i + 2y_v \cos \theta_i \sin \theta_i - 2\rho_i \cos \theta_i] = 0$$

or

$$x_v \Sigma \cos^2 \theta_i + y_v \Sigma \cos \theta_i \sin \theta_i = \Sigma \rho_i \cos \theta_i \qquad (4)$$

and

$$\frac{\partial D^2}{\partial y_v} = \Sigma[2x_v \cos \theta_i \sin \theta_i + 2y_v \sin^2 \theta_i - 2\rho_i \sin \theta_i] = 0$$

or

$$x_v \Sigma \cos \theta_i \sin \theta_i + y_v \Sigma \sin^2 \theta_i = \Sigma \rho_i \sin \theta_i \; . \qquad (5)$$

Let

$$a = \Sigma \cos \theta_i \sin \theta_i$$

$$b = \Sigma \cos^2 \theta_i$$

$$c = \Sigma \sin^2 \theta_i$$

$$d = \Sigma \rho_i \cos \theta_i$$

$$e = \Sigma \rho_i \sin \theta_i \; .$$

Using a, b, c, d, and e in Eqs. 4 and 5, we obtain

$$bx_v + ay_v = d$$

$$ax_v + cy_v = e$$

Thus, the best-fit vertex (x_v, y_v) is given by

$$x_v = \frac{dc - ae}{bc - a^2} \tag{6}$$

$$y_v = \frac{be - ad}{bc - a^2} . \tag{7}$$

The above solution is not valid in the case of parallel straight line segments, i.e., when $\theta_1 = \theta_2 = \ldots = \theta_M$.

REFERENCES

1. A. Rosenfeld and A. Kak, Digital Picture Processing, (Academic Press, New York, 1976).

2. D.E. Knuth, The Art of Computer Programming, Volume 1 (Addison-Wesley Publishing Company, Menlo Park, California, 1973).

3. R.O. Duda and P.E. Hart, Pattern Classification and Scene Analysis (John Wiley & Sons, New York, 1973).

4. M.H. Hueckel, "A Local Visual Operator Which Recognizes Edges and Lines," Journal ACM, 20, 4, October 1973, pp. 634-647.

5. R.O. Duda and P.E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," Comm. ACM 15, 1, January 1972, pp. 11-15.

6. S.A. Dudani and C. Clark, "Vertex-Based Model Matching," Proc. Symp. on Current Mathematical Problems in Image Science, November 1976, Monterey, California.

7. C.T. Zahn, Jr., "An Algorithm for Noisy Templete Matching," Information Processing 74 (North Holland Publishing Co., 1974).

8. C.R. Brice and C.L. Fennema, "Scene Analysis Using Regions," Artificial Intelligence 1, Fall 1970, pp. 205-226.

9. R. Ohlander, "Analysis of Natural Scenes," Ph.D. thesis, Carnegie-Mellon University, Pittsburgh, Pennsylvania, August 1975.

10. K. Price, "Change Detection and Analysis in Multi-Spectral Images," Ph.D. thesis, Carnegie-Mellon University, Pittsburgh, Pennsylvania, December 1976.

11. S.A. Dudani, "Region Extraction Using Boundary Following," Pattern Recognition and Artificial Intelligence (Academic Press, Inc., 1976).

12. R.M. Haralick, K. Shanmugan, and I. Dinstein, "Textural Features for Image Classification," IEEE Trans. Sys., Man, and Cyber. SMC-3, 6 (1973) pp. 610-622.

13. S.A. Dudani, "Aircraft Identification by Moment Invariants," IEEE Trans. Computers C-26, 1, January 1977, pp. 39-46.

14. R. Nevatia, Computer Analysis of Scenes of 3-Dimensional Curved Objects (Birchauser-Verlag, Basel, Switzerland, 1976).